UNIVERSITÀ DI PISA

INTERFACOLTÀ : ECONOMIA - SCIENZE MM. FF. NN.

CORSO DI LAUREA IN "INFORMATICA PER L'ECONOMIA E PER L'AZIENDA"

TESI DI LAUREA

IL PROBLEMA DELL'ESTRAZIONE-TRASFORMAZIONE-CARICAMENTO DI DATI IN UN DATA WAREHOUSE: "MAKE-OR-BUY"

Relatore: **Prof. Nicola Ciaramella**

Relatore: **Franco Farinati**

Candidato: **Andrea Malfitano**

Anno Accademico 2008-2009

Riassunto

Obiettivo della tesi è il benchmark fra due diverse metodologie adottabili per l'implementazione degli estrattori dati utilizzati per l'alimentazione periodica di un data warehouse.

L'indagine sarà facilitata dall'applicazione di entrambe ad un caso reale, ovvero il caricamento dei cubi relativi all'analisi delle ore lavorate su commessa e degli acquisti delle materie prime che servono per la realizzazione delle stesse, utilizzando come punto di partenza i flussi informativi di un'azienda che è stata selezionata come caso di studio.

Durante lo svolgimento del progetto verrà analizzata l'implementazione dei flussi ETL tramite lo sviluppo di interfacce di Oracle Data Integrator, uno dei software di integrazione dati presenti sul mercato, e di programmi scritti in PL/SQL. Questo permetterà di effettuare un confronto critico di tipo tecnologico fra le due soluzioni, basato su diversi tipi di indicatori quantitativi-qualitativi; sarà inoltre sviluppato un paragone di tipo economico per valutare in quali casi sia vantaggioso utilizzare l'approccio della programmazione tradizionale e in quali invece sia preferibile l'acquisto del software.

Indice

Introduzione

Il presente lavoro nasce dalle osservazioni maturate in una serie di progetti riguardanti la migrazione dei dati fra sistemi diversi. L'esperienza accumulata ha portato a riflettere sulla varietà di strumenti che possono essere utilizzati per giungere al medesimo risultato e a interrogarsi su quale fra questi, a seconda delle caratteristiche e della complessità del progetto, conduca ad un'ottimizzazione dei risultati contemporaneamente all'ottenimento di una minimizzazione dei costi sostenuti per perseguirlo.

Obiettivo della tesi sarà quello di valutare pregi e difetti di due modalità differenti di implementazione di estrattori dati alimentanti un data warehouse. Nel primo caso il problema verrà affrontato in modalità tradizionale, ovvero mediante lo sviluppo di programmi/script in PL/SQL specifici per eseguire ognuna delle tre fasi di migrazione, mentre la seconda strada proporrà l'utilizzo per lo stesso scopo di un prodotto di mercato pensato e realizzato per la problematica ETL.

Per la componente pratica del progetto verrà analizzato, come punto di partenza, un caso reale, in cui la problematica di ETL è stata già affrontata in modalità tradizionale. Selezionando un insieme significativo di flussi informativi di input (vengono utilizzati quelli legati al ciclo passivo), si affronterà il problema della realizzazione degli estrattori tramite l'utilizzo del prodotto "Oracle Data Integrator". Obiettivo del progetto sarà anzitutto l'identificazione degli indicatori qualitativi e quantitativi di confronto, che permetteranno in un secondo momento l'analisi di pregi e difetti di entrambi gli approcci. Verranno prese in considerazione, oltre al caso in cui si effettui il primo caricamento del Data Warehouse, anche situazioni di richiesta da parte dell'azienda di modifica delle informazioni in esso caricate, a seguito di un cambiamento

delle esigenze di business (si pensi al caso in cui venga introdotta una nuova dimensione nell'analisi del processo Vendite).

In base ai parametri presi in considerazione quali elementi di confronto, sarà inoltre realizzata una breve indagine di analisi economica per valutare in quali casi (modello generale di complessità del progetto, numero di estrattori, periodo di manutenzione previsto) sia economicamente vantaggioso utilizzare l'approccio tradizionale ed in quali l'approccio tramite sfruttamento del software. Per determinare dei dati quantitativi ben precisi sui risultati del paragone verrà utilizzato l'indice ROI (return on investment), una misura cui in genere si fa ricorso proprio per decidere se è utile investire delle risorse finanziarie per un acquisto e per definire quale potrebbe essere il ritorno per l'investimento nel lungo periodo. Tipicamente il ROI è utilizzato per decisioni di calcolo del ritorno economico per beni più "tangibili" di un software (es. acquistare un nuovo impianto che permetterebbe il miglioramento delle performance della produzione di un prodotto, o tenersi l'impianto vecchio senza spendere i soldi per l'acquisto del nuovo, rinunciando all'incremento delle prestazioni?); ad ogni modo, si tenterà di applicare le tipiche operazioni del calcolo del ROI al caso specifico.

In base a quanto detto, viene ora mostrato l'elenco dei capitoli che fanno parte della tesi e, per ognuno, si da una breve anticipazione dei temi che saranno trattati al loro interno:

- **Capitolo 1 - Motivazioni, requisiti e strumenti:** in questo capitolo verrà presentata l'azienda che ha richiesto la consulenza per il processo di ETL, gli strumenti e i linguaggi utilizzati durante lo sviluppo del progetto. Il paragrafo iniziale sarà dedicato ad un accenno alle tematiche della business intelligence;

- **Capitolo 2 - La progettazione funzionale:** in questo capitolo si esaminerà il progetto dal punto di vista funzionale, ovvero si descriveranno, senza ancora entrare nel dettaglio tecnico, i flussi tramite i quali verrà effettuato il caricamento dei cubi;

- **Capitolo 3 - ETL con Oracle Data Integrator:** in questo capitolo si mostrerà l'utilizzo del software Oracle Data Integrator per l'imple-

mentazione tecnica dei flussi ETL che alimenteranno i cubi del data warehouse;

- **Capitolo 4 - ETL con PL/SQL:** in questo capitolo saranno accuratamente descritti i listati in PL/SQL che permettono il raggiungimento dello stesso risultato;
- **Capitolo 5 - Benchmark: paragone tecnico:** in questo capitolo si effettuerà il paragone delle due soluzioni dal punto di vista tecnico. Verranno introdotti gli indicatori che guideranno nel confronto e nella determinazione di pregi e difetti di ciascun procedimento;
- **Capitolo 6 - Benchmark: paragone economico:** in questo capitolo si effettuerà il paragone delle due soluzioni dal punto di vista economico. Saranno individuati i casi in cui sarà conveniente lo sviluppo in PL/SQL e quelli in cui sarà invece maggiormente auspicabile l'acquisto del software di data integration;

Capitolo 1

MOTIVAZIONI, REQUISITI E STRUMENTI

Nei primi paragrafi di questo capitolo, che precede le parti più "tecniche" del lavoro, si presenta innanzitutto il contesto in cui si inquadrano le operazioni di implementazione degli estrattori, descrivendo dapprima le tematiche della business intelligence, e poi l'impresa che è stata selezionata quale caso di studio. In seguito saranno analizzati il pacchetto software e il linguaggio di programmazione utilizzati per lo sviluppo tecnico dei flussi che consentono il caricamento dei cubi, ovvero i due strumenti adottati da confrontare.

1.1 ETL per la business intelligence

Il lavoro propone la problematica del confronto tra due modalità di implementazione degli estrattori per il caricamento e l'alimentazione di un data warehouse. In questo paragrafo verranno illustrati nel dettaglio i passi di questo importante processo, a partire dalle motivazioni che suggeriscono la memorizzazione di un particolare insieme di dati trasformati, in una struttura diversa dal tipico "contenitore di tabelle" che è il database.

E' verso la metà degli anni 90' che si inizia a sentir parlare di data warehouse. In quegli anni si cerca di studiare un sistema efficiente per fornire ai dirigenti aziendali uno strumento idoneo ad eseguire valutazioni sull'insieme dei dati archiviati nei vari database presenti nelle aziende stesse, finalizza-

ti al supporto dei processi decisionali. Le informazioni direzionali, di cui si necessita, devono essere organizzate per temi o fatti e fornire degli indicatori atti a misurare e valutare le performance aziendali di alcuni processi secondo prospettive rilevanti (dimensioni). Il livello di sintesi dell'informazione da presentare ai manager deve essere molto alto; la sintesi deve però essere in grado di mettere in luce le criticità e, una volta che queste sono state individuate, rendere possibile l'approfondimento del dettaglio (tramite le gerarchie).

Per soddisfare queste esigenze, da quel momento, i dati storici vengono integrati con dati provenienti da fonti esterne e organizzati opportunamente in un altro tipo di base di dati, detta data warehouse, gestita separatamente mediante un tipo di sistema informatico direzionale che metta a disposizione strumenti con interfacce molto curate per consentire agli utenti di eseguire facilmente analisi di dati interattive. Mentre una base di dati operazionale è aggiornata tempestivamente ad ogni verificarsi di un evento aziendale, il data warehouse viene aggiornato periodicamente con nuovi dati storici consolidati e/o aggregati con dati di provenienza esterna ai sistemi informativi operativi. Il memorizzare i dati oggetto dell'analisi in un sistema dedicato permette inoltre di scongiurare problemi prestazionali sui sistemi operativi.

In linea puramente teorica, la base di dati operativa che un'organizzazione usa per le proprie attività quotidiane ed il data warehouse potrebbero coincidere. In pratica questo non avviene per diversi motivi:

- **Diversità delle informazioni:** la base di dati ed il data warehouse contengono in generale informazioni diverse; per esempio, mentre la base di dati contiene lo stato attuale delle applicazioni gestionali ed operative, il data warehouse contiene informazioni storiche che riguardano le applicazioni eseguite in un certo intervallo di tempo; più in generale, non conviene appesantire la base di dati inserendovi informazioni che sono di interesse solo per il supporto alle decisioni e non per le attività gestionali, nè conviene inserire nel data warehouse informazioni che non si è interessati ad analizzare per valutare i processi aziendali.

- **Diversità dei sistemi:** un DBMS ottimizzato per applicazioni gestionali e operative non è necessariamente il più adatto per le applicazioni

di supporto alle decisioni, e viceversa.

- **Conflitto tra le applicazioni:** non conviene fare eseguire allo stesso sistema sia le transazioni che le analisi dei dati perchè queste ultime sono in genere molto costose dal punto di vista computazionale e possono degradare le prestazioni dell'esecuzione delle prime.

Le principali differenze fra le classiche applicazioni transazionali interattive che usano basi di dati (On Line Transaction Processing, OLTP) e le applicazioni per il supporto alle decisioni che usano data warehouse (On Line Analytical Processing, OLAP) sono dettagliatamente riassunte nella figura 1.1. [Albano 2003]

	OLTP	OLAP
Utenti	Impiegati o tecnici	Dirigenti
Scopi	Operazioni giornaliere	Supporto alle decisioni
Organizzazione	Per settori	Per soggetti
Usi	90% predefiniti	90% estemporanei
Dati	Attuali, dettagliati, relazionali, specifici	Storici, sintesi, multidimensionali, integrati
Tipi di accessi	Brevi transazioni con letture e scritture	Letture con ricerche complesse
Numero utenti	Migliaia	Decine
Quantità di dati	100MB-1GB	100GB-1TB

Figura 1.1: Differenze fra applicazioni OLAP e OLTP

Nei sistemi direzionali le informazioni sono generalmente rappresentate attraverso degli ipercubi multidimensionali. Gli assi, ortogonali tra loro, del cubo rappresentano le dimensioni di analisi e all'incrocio dei valori di ogni asse si trovano i fatti che sono oggetto dello studio. Nel caso classico delle vendite, ad esempio, si possono considerare le dimensioni di analisi: data, negozio e prodotto. Gli elementi dell'ipercubo rappresentano quindi la vendita di un determinato prodotto in una certa data e in un particolare negozio.

I valori della singola dimensione possono essere strutturati sotto forma di gerarchie. Ad esempio se l'asse temporale individua delle date, queste possono essere raggruppate per settimane o mesi, i mesi possono essere a loro

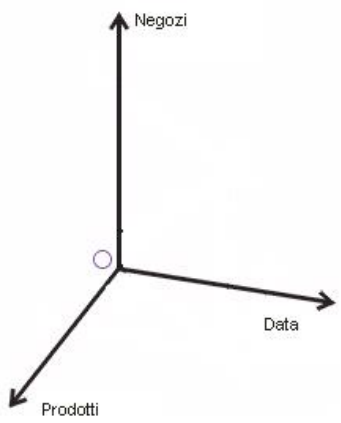


Figura 1.2: Esempio di ipercubo

volta raggruppati in quadrimestri, e così via fino agli anni, decenni, secoli. Analogamente la dimensione spaziale può essere strutturata con una gerarchia del tipo strada, quartiere, comune, provincia, regione, nazione. Il tipo di raggruppamento da utilizzare dipende ovviamente dal problema che si sta analizzando.

Per navigare all'interno dell'ipercubo multidimensionale, sono utilizzabili le seguenti operazioni:

- L'operazione di roll-up consente di sintetizzare progressivamente i fatti in base alla gerarchia di valori di una dimensione fino a farla sparire. Nell'esempio delle vendite, rappresentando in una tabella la dimensione temporale sulle righe e i prodotti sulle colonne, si possono raggruppare i dati giornalieri in modo che le nuove righe contengano i totali mensili per ogni prodotto. Raggruppando ulteriormente si può arrivare a comprimere la dimensione temporale ottenendo un'unica riga con le vendite totali.

- Quella di drill-down è l'operazione inversa del roll-up. Permette di espandere i dati aggregati ad un certo livello di gerarchia di una dimensione, nei dati più dettagliati del livello gerarchico sottostante. Ad esempio si possono esplodere i dati di vendita mensili in quelli giornalieri.

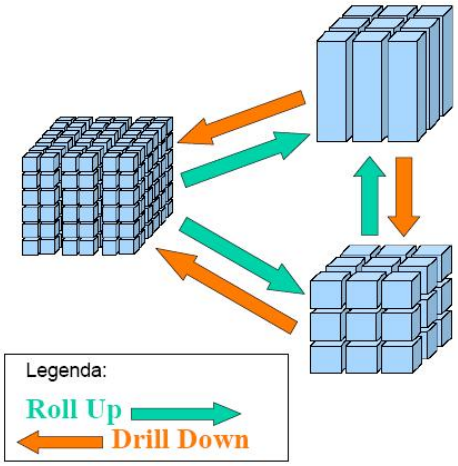


Figura 1.3: Le operazioni di roll-up e drill-down

- Le operazioni slice e dice consentono di filtrare i dati dell'ipercubo fissando un valore su una o più dimensioni. Quindi, per esempio, fissando un negozio lungo la dimensione geografica, è possibile ottenere la fetta di cubo che contiene tutte le vendite in ogni giorno di tutti i prodotti venduti in quel determinato negozio. Applicando iterativamente il processo sulle varie dimensioni si arriva a individuare il singolo fatto.

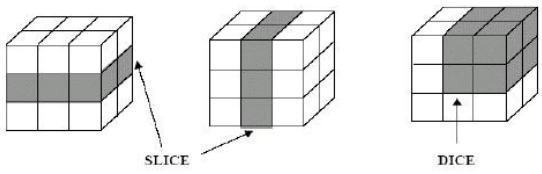


Figura 1.4: Le operazioni di slice e dice

- L'operazione di pivoting consente di analizzare le stesse informazioni da prospettive diverse, ruotando il cubo lungo una direzione. Se una tabella rappresenta l'andamento delle vendite nel tempo dei prodotti nei negozi, è possibile estrarne una che contenga quanto hanno venduto nei vari anni i singoli negozi per ogni tipologia di prodotto.

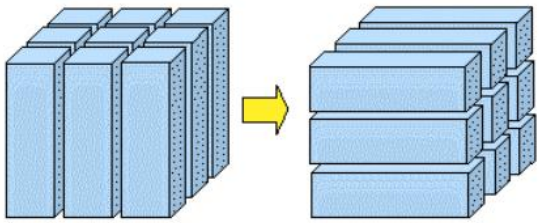


Figura 1.5: L'operazione di pivoting

È proprio in questo contesto, oltre che in altre situazioni di integrazione di dati o di migrazione degli stessi fra diversi database, che trovano spazio le applicazioni ETL il cui scopo è rendere disponibili i dati raccolti in azienda, provenienti da fonti eterogenee, ai soggetti incaricati di assumere le decisioni, nella forma e secondo le tempistiche più idonee a supportare il processo decisionale. Per riuscire a sfruttare pienamente i dati operativi disponibili, l'infrastruttura tecnica preposta al sostegno dei processi decisionali deve essere in grado di raccogliere, consolidare, trasformare e trasferire i dati, predisponendoli al meglio per la successiva fase analitica.

Il processo ETL si compone di tre fasi principali:

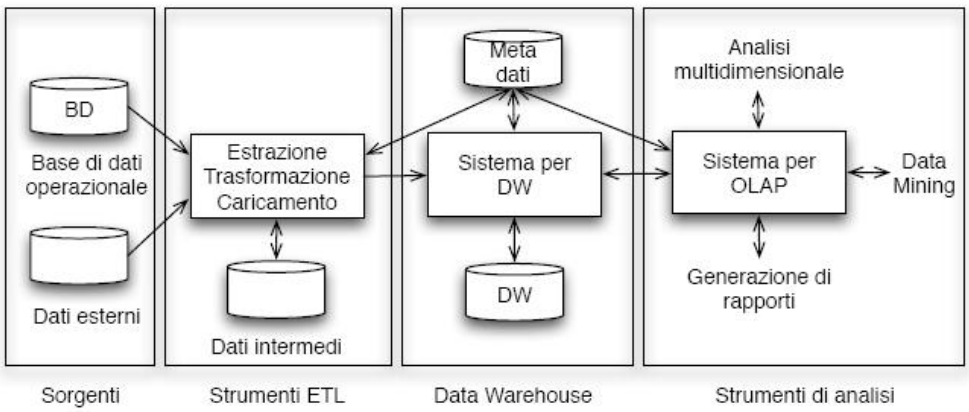


Figura 1.6: Il processo di Data Warehousing

- **Estrazione:** questa fase implica la predisposizione di procedure in gra-

do di leggere i record contenuti tanto in diversi database quanto da file sorgenti di diversi formati, e di predisporli per il successivo stage di trasformazione.

- **Pulizia e Trasformazione:** questa fase consiste nel trasformare, pulire e integrare i dati. Con l'operazione di pulizia si migliora la qualità dei dati che verranno caricati nel data warehouse correggendo gli errori e le inconsistenze presenti nei dati elementari (ad es. dati duplicati, dati mancanti, valori errati o nulli). Sono infatti numerosi gli errori di battitura che sono difficilmente evitabili se non si prevedono procedure di controllo dei valori inseriti; altrettanto comune è riscontrare differenze di codifica dei dati nello stesso campo (ad es. utilizzare abbreviazioni differenti per indicare lo stesso comune o la stessa nazione). I tipi di trasformazione che possono essere eseguiti possono essere classificati in quattro raggruppamenti [Dotnethell 2009]:

  - Business Intelligence Transformations: permettono di implementare logiche di business sui dati.
  - Row Transformations: consentono la creazione e l'aggiornamento di colonne in base alla riga passata in input al task (ad esempio Colonne derivate, Script component, Conversioni di date, Comandi OLEDB per riga).
  - Rowset Transformations: consentono la creazione di set di dati aggregati, ordinati, trasposti (ad esempio il PIVOT/UNPIVOT, i task di raggruppamento, il task sort).
  - Split and Join Transformations: consentono la distribuzione di un input in più output, l'unione di più input in un output oppure eseguono operazioni di lookup (multicast, union e lookup task).

- **Caricamento:** rappresenta l'ultima fase del processo ETL e consiste nel caricamento delle informazioni nella base di dati o nel data warehouse di destinazione. Durante questa fase il programmatore si trova ad affrontare due problematiche che solo all'apparenza risultano semplici. La prima riguarda la scelta dell'impostazione del tipo di caricamento,

ovvero se il caricamento nella base di dati deve avvenire in forma periodica oppure in forma continuativa. Quest'ultima opzione risulta spesso costosa poiché richiede reti dedicate ad alta velocità. L'altra problematica riguarda la scelta tra due modelli di replicazione dei dati, ovvero di tipo push in cui l'applicativo spinge i dati trasformati verso la base di dati di destinazione e di tipo pull in cui, al contrario, l'applicazione o la base di dati richiedono i dati in conformità alle specifiche esigenze del momento.

Tutte queste operazioni, dall'estrazione alle varie trasformazioni, possono essere realizzate attraverso lo sviluppo di programmi/script, utilizzando linguaggi di programmazione tradizionali, nel caso prescelto il PL/SQL, oppure attraverso l'utilizzo di un tool appositamente pensato per questo scopo, nel caso prescelto Oracle Data Integrator (ODI).

1.2 Siram ed il Progetto Erasmus

Siram è la prima società italiana nel settore dei multiservizi tecnologici. Opera dal 1912 in tutta Italia nel settore della gestione integrata dell'energia per gli enti pubblici e l'industria con forti capacità tecniche che le permettono di offrire servizi mirati alle diverse realtà territoriali a cui si rivolge. Nel corso degli anni ha sviluppato esperienze significative nel campo del Global Service e del Facility Management, acquisendo competenze specialistiche che arricchiscono il pacchetto di offerta rivolto ai Clienti. Siram è nata il 1° ottobre 2002 dalla fusione tra Siram S.p.A. e Dalkia S.p.A. Il Gruppo Dalkia, di cui fa parte, è leader in Europa nel settore dei servizi energetici ed è presente in 38 paesi con più di 48.000 dipendenti. E' partecipata da Veolia Environnement e da EDF. Il gruppo e le due Azioniste sono N°1 mondiale dei servizi per l'ambiente: Energia (Dalkia), Trasporti (Veolia Trasporti), Pulizia e Rifiuti (Veolia Servizi Ambientali), Acqua (Veolia Acqua), con un fatturato complessivo di 28,6 mld di Euro (consolidato 2006 Veolia) e oltre 270.000 collaboratori in tutto il mondo.

Siram ha scelto Business Reply quale partner per la realizzazione del nuovo sistema informativo aziendale basato sulle Oracle E-Business Suite. L'obiettivo di Siram è stato quello di dotarsi di un supporto flessibile ed integrato per perseguire le strategie di business aderendo agli standard definiti dalla casa madre francese e favorendo l'interscambio di informazioni con la stessa.

All'inizio del 2006, Dalkia International ha esteso lo scopo originario del progetto eBS e dato vita al progetto ERASMUS, allargando lo studio e l'implementazione della soluzione anche alle filiali di Spagna e Portogallo. Il progetto, nella sua configurazione definitiva, ha coinvolto tre sussidiarie italiane, otto spagnole e cinque portoghesi.

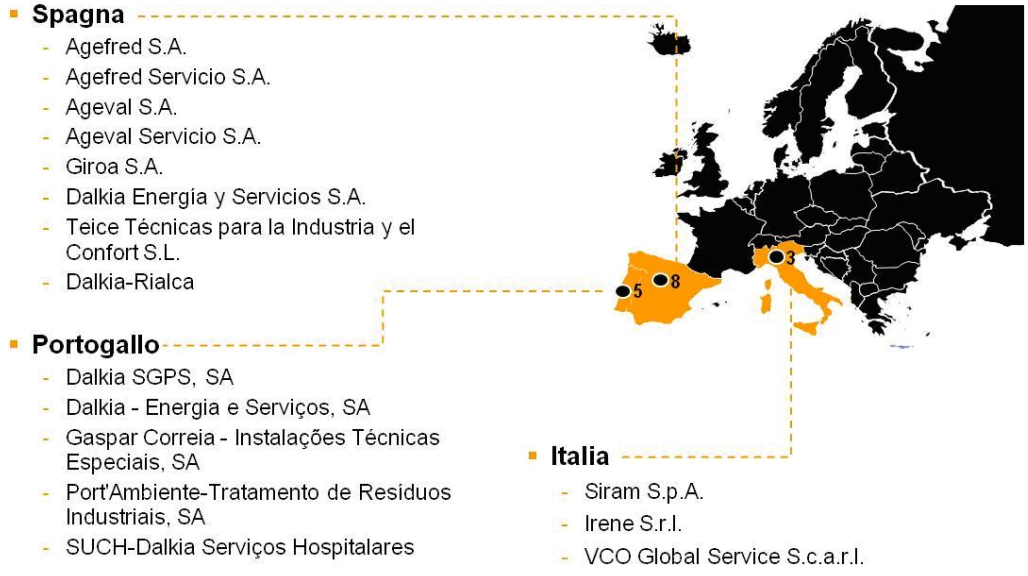


Figura 1.7: Imprese aderenti al progetto

Dalkia, con il contributo di Reply, si è quindi posta e raggiunto l'obiettivo ambizioso della realizzazione di un sistema informativo a supporto dei processi aziendali trasversali (Amministrazione e Finanza, Controllo di Gestione, Servizi generali) e dei processi di core-business, integrando anche i sistemi verticali a supporto delle operation ed i sistemi informativi specifici delle singole country, garantendo al tempo stesso agili-

tà operativa su base locale e omogeneità dei processi in conformità ai modelli definiti dalla casa madre.

Alla base del rinnovamento del Sistema Informativo di Siram/Dalkia è stata rilevata la necessità di:

- Sostituire i sistemi attuali basati sulla piattaforma gestionale AS/400 (Italia) e Modulix (Spagna e Portogallo) che hanno subito numerose personalizzazioni e che sono caratterizzati da limitazioni funzionali e strutturali.
- Progettare e costruire un Sistema Informativo integrato che consentisse la flessibilità necessaria per supportare al meglio la realizzazione dalle strategie di business di Siram (anche in relazione al programma "Dalkia 2010").
- Aderire agli standard definiti dalla Casa Madre Francese in materia di "Core System" e quindi favorire l'integrazione e l'interscambio di informazioni con la stessa.
- Assicurare un'adeguata evoluzione funzionale capitalizzando sulle esperienze maturate nel tempo e sfruttando le economie di scala generate dal perimetro non più locale di una singola Country.
- Permettere di strutturare le funzioni aziendali per Processi garantendo una maggiore aderenza al Sistema di Qualità Aziendale.
- Garantire la sicurezza dei dati/informazioni gestite, basata su ruoli organizzativi e profili di accesso.
- Assicurare la gestione delle autorizzazioni basate su gerarchie di approvazione legate a "workflow".
- Ottimizzare il rapporto degli Utenti con la Struttura ICT (ottimizzazione delle funzioni di help desk, migliore gestione degli asset ICT, ecc.) strutturando un servizio basato su un primo livello locale ed un secondo livello gestito centralmente da Siram.

Più in dettaglio, la soluzione eBS disegnata da Siram con il contributo di Reply, supporta le aree Finance, Sales, Logistics, Project, EAM/Service. I moduli standard di E-BS sono integrati da moduli applicativi specifici,

realizzati con la tecnologia Oracle Application Framework, per fornire un supporto "su misura" ai processi core del business di Siram. Applicazioni verticali proprietarie Dalkia (es. Energy Management) e sistemi dipartimentali di terze parti sono integrate nella soluzione mediante l'utilizzo di componenti dell'Oracle SOA Suite.

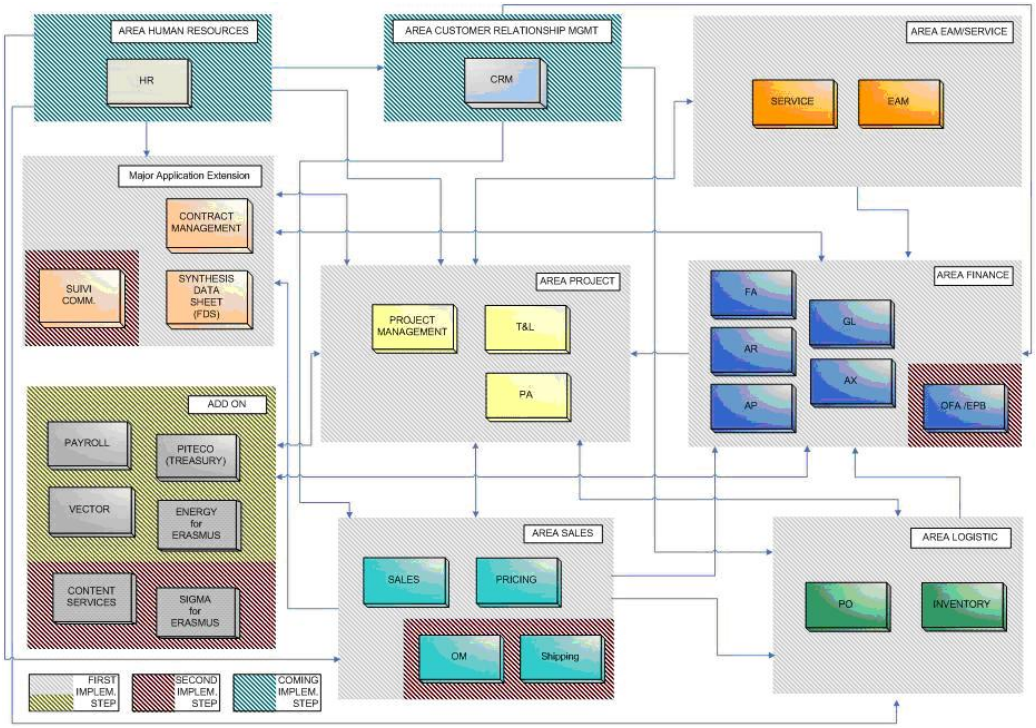


Figura 1.8: Il sistema Oracle

Tra gli obiettivi più significativi che sono stati posti al team di progettazione e realizzazione del progetto Erasmus, oltre alle necessità elencate precedentemente, si possono citare in generale:

- La progettazione di un Sistema Informativo integrato adeguato alle necessità dell'organizzazione aziendale attuale e che supporti la stessa nelle sue evoluzioni futura.
- Favorire l'adozione di un modello di processi comune ed armonizzato per le diverse Filiali (Italia, Spagna e Portogallo) ed Unità di Business (BU tradizionali, Industria e Facility Management).

- Fornire servizi a valore per tutta l'organizzazione Aziendale, trattando efficacemente le informazioni con valenza di interfunzionalità.
- Mettere a disposizione degli Utenti informazioni aggiornate ed online per consentire operazioni efficaci garantendone, al tempo stesso, univocità e consistenza.

Ed in particolare, per le aree di riferimento:

- **Finanza e controllo di gestione**
  - Gestire i processi amministrativi nel rispetto delle normative di legge vigenti a livello Italiano (Normativa Civilistico / Fiscale) Europeo (IFRS) e Americano (SOX).
  - Assicurare un consolidamento contabile, a livello di Gruppo, affidabile e strutturato secondo le regole aziendali definite (riducendo complessivamente il costo di gestione del Bilancio consolidato e delle situazioni gestionali consolidate).
  - Migliorare il processo di Fatturazione attiva.
  - Migliorare il processo di Gestione dei crediti.
  - Ottimizzare la gestione degli interessi moratori (attivi e passivi).
  - Supportare una gestione efficace ed efficiente delle ATI.
  - Migliorare la facilità di reperimento delle informazioni necessarie per attuare un corretto ed ampio controllo di gestione.
- **Acquisti**
  - Gestire in modo integrato i processi di acquisto a partire dalla RdA (garantendo adeguata visibilità degli ordini inseriti a Sistema a vantaggio di altre Aree aziendali).
  - Rendere più efficace il processo di gestione degli acquisti (utilizzo esteso al Gruppo Siram di accordi quadro con relativo monitoraggio e quindi del numero di acquisti gestiti sugli stessi; riduzione delle attività a basso valore aggiunto; adozione di strumenti evoluti di Internet Procurement).

  - Automatizzare i meccanismi di valutazione delle prestazioni dei Fornitori (attraverso una analisi puntuale dei flussi di acquisto) e più in generale favorire la gestione di KPI e benchmarking sull'Area Procurement.
  - Migliorare il processo di gestione e chiusura del Ciclo passivo (anche in ragione dell'utilizzo di un sistema di gestione documentale).

- **Commerciale**
  - Gestire e monitorare il portafoglio commerciale in modo strutturato (unico ambiente per tutte le tipologie di offerte/clienti; utilizzo di strumenti integrati di controllo e reporting capaci di correlare le informazioni commerciali a quelle economiche di contratto).
  - Gestire un paragone più puntuale tra Budget commerciale e Budget realizzato.
  - Gestire le opportunità e le proposte utilizzando uno strumento di tracciatura del ciclo commerciale (utilizzo di sistemi di opportunity management, evidenziazione della fase: richiesta di partecipazione, offerta, aggiudicazione, contrattualizzazione, ecc.)
  - Realizzare l'informatizzazione della scheda di sintesi (unitamente ad una gestione tramite workflow delle autorizzazioni rilevanti).

- **Progettazione**
  - Favorire il trasferimento della documentazione prodotta in sede progettuale alle funzioni operative dopo l'acquisizione è stata acquisita (anche in ragione dell'utilizzo di un sistema di gestione documentale).
  - Migliorare e rendere più rapida e strutturata l'attività di prevenzione relativa alle diverse linee di Business dell'azienda.
  - Permettere la generazione dei ricavi basata su criteri variabili (avanzamento attività, raggiungimento di una specifica data, ad evento, ecc.) e successiva generazione di fatture attive (ad esempio nel caso di progettazioni relative a rapporti in ATI).

* Possibilità di gestire “progetti interni” per assicurare un puntuale controllo dei costi in rapporto al budget fissato.
* Consentire una condivisione controllata di documenti ed informazioni (in fase progettuale) tra soggetti interni ed esterni all’azienda.

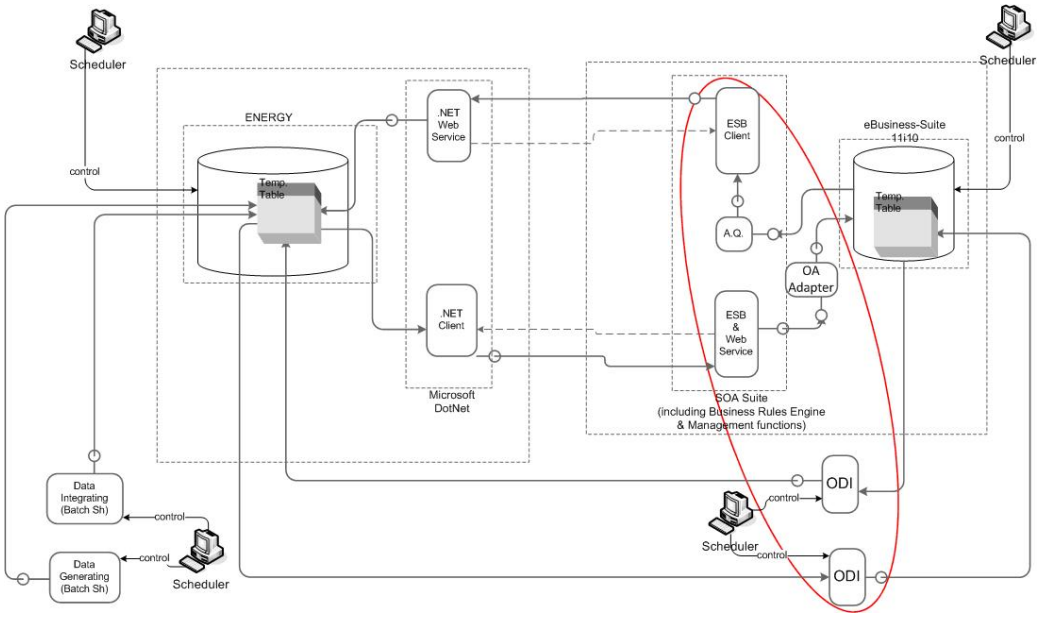


Figura 1.9: Modello per l’integrazione delle E-Business Suite con Energy

Come precedentemente evidenziato, una delle peculiarità e dei punti di attenzione del progetto è stata la realizzazione di una architettura informatica che permettesse di integrare la soluzione E-Business Suite con i sistemi Dalkia (Energy e Sigma); per affrontare tale problematica sono state utilizzate alcune componenti tecnologiche di Oracle SOA suite (ESB - Enterprise Service Bus e ODI - Oracle Data Integrator).

La figura 1.9 illustra il modello di integrazione implementato per l’integrazione delle E-Business Suite con il sistema Energy. La realizzazione delle procedure di migrazione dati e degli estrattori dei dati per l’alimentazione del data warehouse di Siram è stata invece realizzata mediante l’utilizzo di tecniche di programmazione classiche e l’uso del linguaggio PL/SQL.

1.3 PL/SQL

Il PL/SQL è un linguaggio procedurale, estensione dell'ANSI SQL, di proprietà della Oracle, strutturato e a blocchi. Tale linguaggio permette l'interrogazione delle basi di dati e la manipolazione ed estrazione dei dati residenti. Il costrutto di base in PL/SQL è il block (blocco). I blocchi permettono ai programmatori di combinare logicamente i comandi SQL in unità. In un blocco, costanti e variabili possono essere dichiarate, e le variabili possono essere utilizzate per memorizzare i risultati di una query. Le istruzioni in un blocco PL/SQL includono istruzioni SQL, strutture di controllo (loop), istruzioni di condizione (if-then-else), manipolazione delle eccezioni (controllo errori), e chiamate ad altri blocchi PL/SQL.

```
DECLARE
    -- Blocco di dichiarazione (opzionale)
BEGIN
   -- Codice da eseguire
EXCEPTION
   -- Gestione eccezioni(opzionale)
END;

/* Esempi di commenti
multilinea.. */
--commento su singola linea
```

Figura 1.10: Le tre sezioni di un programma PL/SQL

Un blocco è generalmente costituito da tre elementi [Wikipedia 2009]:

- **DECLARE:** questa è la sezione dichiarativa del blocco. In essa vengono pertanto definiti gli elementi utilizzati nel corpo del blocco. Ad esempio variabili, constanti, cursori, array, etc. In questa sezione tali elementi possono essere anche inizializzati. La sezione non è obbligatoria, per cui si potrebbero avere un blocchi PL/SQL in cui non esiste questa sezione perchè non vengono utilizzati nè variabili nè costanti.
- **BEGIN:** questa è l'unica sezione obbligatoria di un blocco PL/SQL, detta anche "corpo". All'interno di essa vengono definiti i comandi che devono

essere eseguiti, le istruzioni SQL, l'assegnazione di valori alle variabili, il controllo del flusso ecc.

- **EXCEPTION:** anche questa sezione è opzionale, ma è buona norma inserirla sempre, in quanto è la parte del blocco destinata alla gestione degli errori. Gli errori gestibili possono essere classificati in errori di default o errori definiti dagli utenti che vogliono un controllo più puntuale sull'esecuzione del codice.

- **END:** questo è il comando che chiude il blocco, e deve sempre essere presente (come il BEGIN). Si precisa che il comando deve essere sempre seguito da un punto e virgola (;). Se ci si dimentica di inserirlo, il blocco non funziona.

I blocchi PL/SQL che specificano procedure e funzioni possono essere raggruppati in package (pacchetti). Un package è simile a un modulo e ha un'interfaccia e un'implementazione a parte. Oracle offre diversi package predefiniti, per esempio, routines di input/output, manipolazione di files, pianificazione di jobs, ecc.

Un'importante caratteristica di PL/SQL, che si cita a conclusione di questo paragrafo di presentazione del linguaggio, è che offre un meccanismo per processare i risultati delle query in un modo orientato alle tuple, il che vuol dire, una tupla alla volta. A questo scopo, vengono utilizzati i cursori. Un cursore è fondamentalmente un puntatore al risultato di una query ed è impiegato per leggere i valori degli attributi delle tuple selezionate, inserendoli in variabili. Un cursore è tipicamente usato in combinazione con un costrutto loop in modo che ogni tupla letta dal cursore possa essere processata individualmente. Nel corso dell'implementazione tecnica, come vedremo in uno dei capitoli successivi, faremo uso di diversi cursori per memorizzare quei soli record che devono essere processati, e di cui si vogliono mantenere i dati.

1.4 I tool di Data Integration

Il mercato degli strumenti di data integration sta conoscendo un momento di sviluppo significativo, dovuto soprattutto all'attenzione al controllo dei co-

sti, che ha reso i tool competitivi dal punto di vista del costo, e alla necessità sempre più comune di dover integrare fonti dati eterogenee, in particolar modo su progetti di complessità medio-alta, rispetto all'implementazione di programmi sviluppati ad hoc per le esigenze dei singoli progetti.

Gartner, famosa società di advisory di riferimento nel campo dell'information technology, in una delle sue ricerche, analizza anche il posizionamento dei tool di data integration delle varie case produtttrici, valutando le prestazioni di ognuno dei tool rilevanti presenti sul mercato. Gartner stima la dimensione del mercato degli strumenti di Data Integration approssimativamente in 1,44 miliardi di dollari alla fine del 2007, e ritiene che tale cifra cresca ogni anno di una percentuale superiore al 17%.

Secondo Gartner, le case produttrici che operano in questo settore si propongono di realizzare software che possiedano un'architettura valida per diversi tipi di scenari di Data Integration, fra i quali:

- ETL per la BI e il data warehousing: l'estrazione di dati da sistemi operazionali, trasformazione e unificazione dei dati provenienti da sorgenti differenti e loro caricamento in strutture che ospitino dati integrati adibiti a scopi di analisi.

- Creazione di master data store integrati: acquisizione, verifica, unicità e integrazione di dati e processi, considerati fondamentali per il successo dell'impresa, in dei master data store seguendo la logica del master data management. Fondamentale risulta anche la definizione di un repository in cui mantenere i metadati relativi alle entità dei data master.

- Migrazioni e conversioni di dati da sistemi legacy: un problema che tradizionalmente veniva risolto implementando programmi specifici, ma che oggi può anche essere risolto tramite l'utilizzo di tool di data integration.

- Sincronizzazione di dati tra applicazioni operazionali: una necessità che richiede agli strumenti di data integration di assicurare la consistenza tra applicazioni al database-level, sia in modo unidirezionale che bidirezionale.

- Creazione di federated view di dati da sorgenti di dati multiple: la data federation, riferita evidentemente all'integrazione di informazioni di imprese differenti, sta crescendo di popolarità come approccio per la realizzazione dell'integrazione real-time di view tra sorgenti di dati multiple, senza un movimento fisico dei dati, e la previsione da parte dei software di funzionalità che permettano la sua realizzazione è ormai sempre più richiesta.

- Unificazione di dati strutturati e non strutturati: uno scenario nuovo e sicuramente in crescita che spinge verso la realizzazione di tools che permettano di fondere dati provenienti da sorgenti di dati strutturati quanto non strutturati.

La ricerca eseguita da Gartner in questo settore è sintetizzata nel "Magic quadrant for data integration tools", poichè Gartner colloca tutti i maggiori venditori di soluzioni di data integration in un "quadrante magico", che mostra in maniera intuitiva le performance del singolo tool rispetto a tutti gli altri. [Gartner 2008]

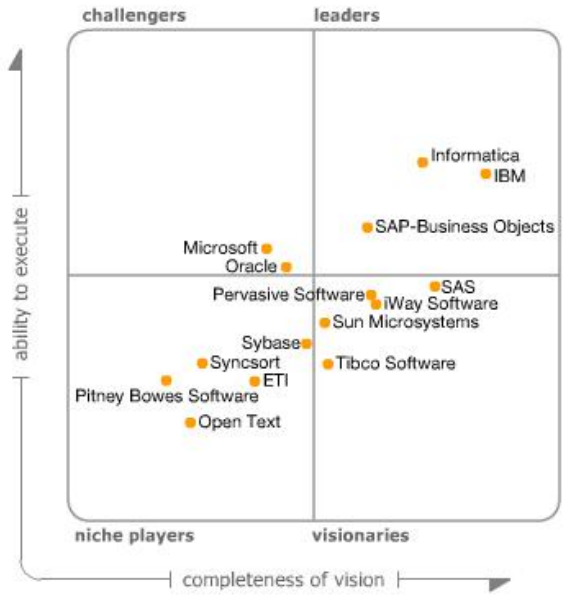


Figura 1.11: Magic quadrant for data integration tools

I due criteri principali che Gartner prende in considerazione per la collocazione di ogni impresa all'interno di una precisa zona del suo quadrante sono:

- **Abilità di esecuzione.** E' una misura della qualità del prodotto e del fornitore, del volume del fatturato e dei servizi offerti ai clienti, principalmente quelli descritti in precedenza. In aggiunta viene enfatizzata la capacità di offerta, da parte del tool, di servizi per la gestione dei metadati;
- **Completezza di visione.** E' una misura dell'abilità dei fornitori nel rispondere ai bisogni attuali e futuri dei clienti. Questo criterio esprime quindi la capacità del venditore nel capire, e se possibile anticipare, le tendenze del mercato e nello realizzare un prodotto innovativo, che dimostri la propria indipendenza rispetto agli altri tool presenti nello stesso settore;

A seconda dei meriti che l'azienda produttrice del tool dimostra di possedere in ambo queste categorie, ad essa viene assegnata l'etichetta di "leader", "challenger", "visionary" o "niche player".

Il tool che verrà utilizzato nell'ambito del processo di estrazione, trasformazione e caricamento del data warehouse, come detto in precedenza, è Oracle Data Integrator. Come è possibile osservare dalla figura 1.8 (che si riferisce alla ricerca dell'anno 2008), la sua azienda produttrice (la Oracle) viene collocata fra i challengers.

Gartner spiega tale decisione fornendo una serie di pregi e difetti dell'offerta dell'impresa nel campo della data integration, che hanno suggerito una sua collocazione in quella zona.

Nel Caso specififico di Oracle Data Integrator sono stati identificati:

- **Punti di forza**
  - Le capacità di data integration di Oracle sono concentrate in due specifici tool: Oracle Warehousew Builder (OWB) e Oracle Data Integrator (ODI), uno strumento stand-alone che appartiene alla famiglia dei prodotti "Oracle Fusion Middleware". Recentemente,

inoltre, Oracle ha rilasciato la “Data Integration Suite”, che si basa su ODI ma include una serie di nuove componenti. L’acquisizione da parte di Oracle di “Bea Systems” incrementa le possibilità di un’espansione delle sue capacità di data federation.

- L’adozione sia di OWB che di ODI continua a crescere, soprattutto in relazione ad esigenze di BI e di data warehousing. I clienti sono soddisfatti delle complete funzionalità di ETL di questi tool, di integrazione con il DBMS Oracle (nel caso di OWB), di integrazione con gli altri componenti e le applicazioni della famiglia Oracle Fusion Middleware nel caso di ODI) e indicano la vasta presenza di Oracle e la sua forza sul mercato globale come principali ragioni per scegliere questi strumenti.
- Oracle distingue i suoi tool dagli altri presenti nel suo stesso mercato, realizzando funzionalità di embedded ETL attraverso l’utilizzo congiunto di Oracle Applications, e degli strumenti per MDM e la BI. Ciò consentirà di aumentare il tasso di adozione da parte dei clienti per le varie tecnologie di Oracle. Oracle ha iniziato a tentare l’integrazione fra OWB e ODI (ad esempio introducendo l’utilizzo dei knowledge module di ODI per i tipi target non-Oracle dentro OWB) e afferma che sta lavorando per offrire un unico set di strumenti di integrazione dei dati nel lungo periodo. Il primo passo consiste nella realizzazione di un unico ambiente di design e nella possibilità di interoperabilità runtime fra i due strumenti.

- **Punti di debolezza**

  - Oracle ha offerto una gamma di funzionalità che incontri le necessità e le richieste del mercato, con un particolare accento sull’ETL tradizionale, invece che istituire una visione globale e innovativa per il futuro degli strumenti di integrazione dei dati e il loro ruolo nel contesto più ampio della gestione delle informazioni. Tuttavia questo approccio pragmatico al mercato produrrà una crescita sostanziale dell’utilizzo delle sue tecnologie e una vasta presenza sul mercato nel lungo periodo.

– Nonostante il dichiarato orientamento verso l'unificazione di OWB e ODI per offrire un unico strumento integrato, il portafoglio di Oracle resta frammentato in vari prodotti. Questo può condurre a confusione per i clienti su come la funzionalità desiderata sia collegato ed integrata. Per Oracle il miglioramento della progettazione delle architetture software dei suoi ambienti e la riduzione del numero di repository di metadati sottostanti le varie tecnologie offerte sarà fondamentale per il suo successo a lungo termine e per l'incremento del customer value.

1.5 Oracle Data Integrator

Terminata la panoramica sui tool di data integration verrà ora illustrata l'architettura di Oracle Data Integrator, tool che verrà utilizzato per la realizzazione del processo di ETL.

L'architettura di Oracle Data Integrator è organizzata attorno a un repository centrale, che viene acceduto in modalità client-server da moduli grafici e agenti di esecuzione, scritti interamente in Java. [Oracle 2006]

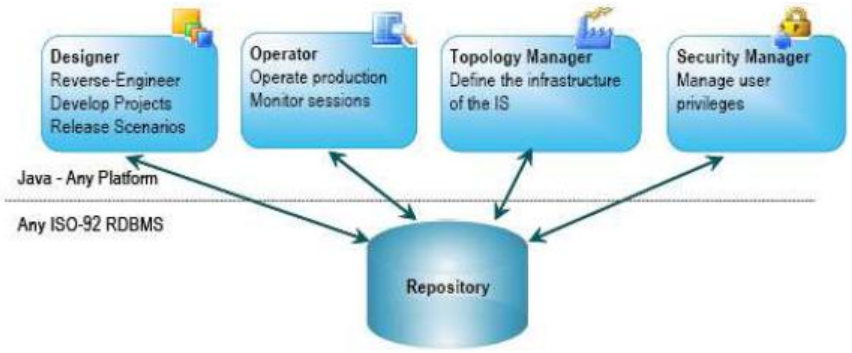


Figura 1.12: I moduli grafici connessi al repository

I moduli grafici sono:

- **Il Designer,** che definisce le regole dichiarative per la data trasformation e la data integrity. Tutto lo sviluppo del progetto prende posto in

questo modulo; esso è in pratica il core module per gli sviluppatori e gli amministratori dei metadati.

- **L'Operator,** che gestisce e monitora l'esecuzione dei progetti. Esso mostra i log di esecuzione, il numero di righe processate, statistiche di esecuzione, ecc. Può quindi essere utilizzato anche per scopi di debugging.
- **Il Topology Manager,** che definisce la struttura logica e fisica dell'infrastruttura. Server, schemi e agenti sono memorizzati nel master repository attraverso questo modulo, generalmente dall'amministratore.
- **Il Security Manager,** che gestisce i profili degli utenti e il loro privilegi di accesso. Tale modulo è quindi soprattutto utilizzato dall'amministratore della sicurezza.

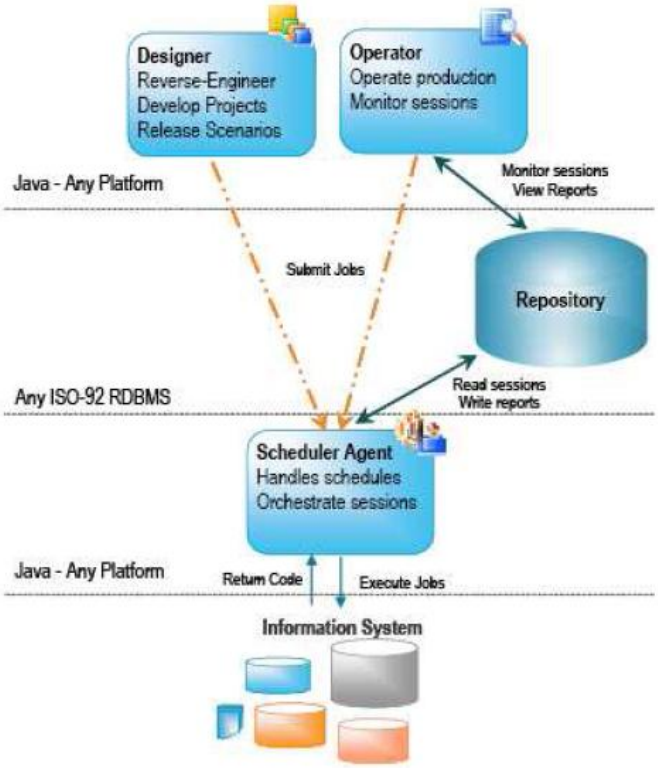


Figura 1.13: I componenti di runtime

Tutti i moduli grafici memorizzano le informazioni nel repository centrale.

Durante il runtime, lo Scheduler Agent coordina l'esecuzione degli scenari. Esso può essere installato su qualunque sistema operativo che supporti una Java Virtual Machine, e la sua esecuzione viene generalmente lanciata da uno dei moduli grafici. Nella particolare architettura di Oracle Data Integrator, lo Scheduler Agent va difficilmente a realizzare esso stesso delle trasformazioni; si limita infatti semplicemente a richiamare il codice necessario dal repository e far richiesta al database server o al sistema operativo affinchè esso venga eseguito. Quando ne viene completata l'esecuzione, lo Scheduler Agent aggiorna i log di esecuzione nel repository e infine riporta statistiche e eventuali log di errore.

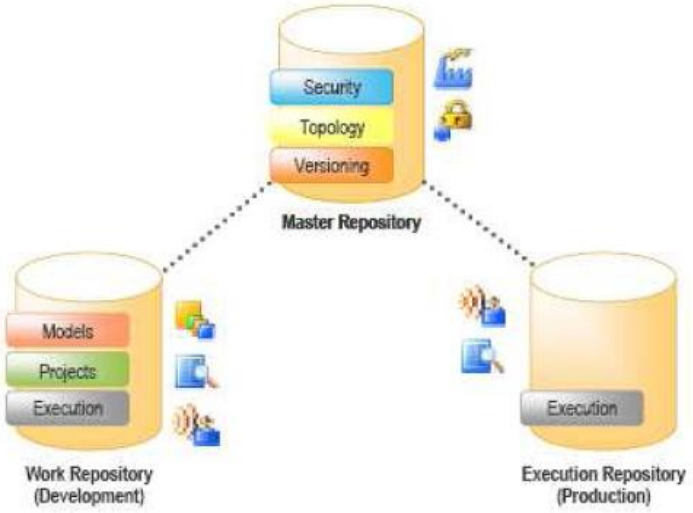


Figura 1.14: Master repository e work repository

Il repository centrale è costituito da un master repository e diversi work repository. Tali repository sono database memorizzati in un DBMS relazionale e vengono acceduti in modalità client-server dai vari componenti dell'architettura. Solitamente esiste un solo master repository, che contiene le informazioni di sicurezza, le informazioni sulla topologie e le versioni degli oggetti. Le informazioni contenute nel master repository sono mantenute grazie al Topology Manager e al Security Manager. Gli oggetti del progetto vengono invece mantenuti nel work repository, perciò possono logicamente coesisterne diversi nella stessa installazione; ciò è particolarmente utile per mantenere ambienti

separati fra i progetti.

Gli utenti gestiscono il contenuto di un work repository attraverso i moduli Designer e Operator, ed essi vengono inoltre interrogati dall'Agent durante il runtime. Quando un work repository memorizza informazioni relative soltanto all'esecuzione, esso è definito execution repository.

Nel caso specifico dell'ETL, Oracle Data integrator propone un approccio del tutto nuovo rispetto alle soluzioni tradizionali. Solitamente i tool ETL operano innanzitutto estraendo i dati dalle varie sorgenti, poi trasformando i dati estratti su un motore ETL middle-tier e quindi caricando i dati trasformati nel data warehouse target. In questo caso il motore ETL realizza la data transformation, e talvolta il controllo della qualità dei dati, secondo una logica riga a riga; questo può portare facilmente alla formazione di colli di bottiglia durante l'intero processo. In aggiunta i dati devono essere spostati sulla rete due volte: la prima dalle sorgenti al server ETL e la seconda dal server ETL verso il data warehouse di destinazione.

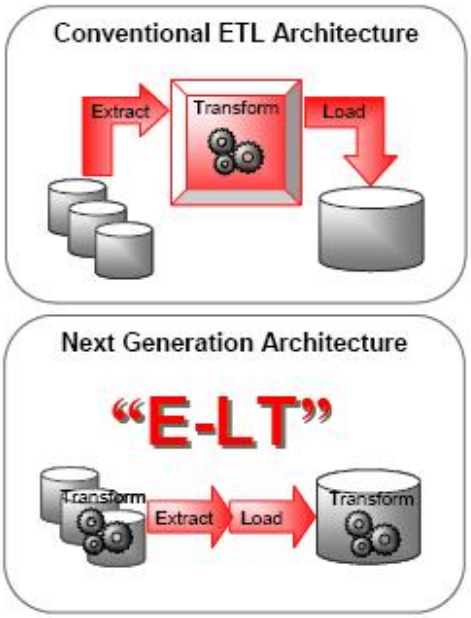


Figura 1.15: Architettura ETL VS Architettura E-LT

Oracle Data Integrator propone invece un'architettura di tipo E-LT: in so-

stanza rialloca il luogo della data transformation nel RDBMS che gestisce le tabelle target. Ciò elimina la necessità di un server ETL stand-alone e del motore e invece sfrutta la potenza del RDBMS.

Si mostrano di seguito, a conclusione del paragrafo introduttivo su Oracle Data Integrator, le key reasons che hanno spinto più di 500 imprese nel mondo a scegliere questo strumento per il loro processi di ETL:

- Sviluppo e gestione dei progetti più semplice e veloce: L'approccio orientato alle regole dichiarative riduce notevolmente la curva di apprendimento del prodotto e aumenta la produttività degli sviluppatori, facilitando inoltre la manutenzione. Questo approccio permette di separare le regole dichiarative (il "cosa"), dai data flow (il "come").

- Realizzazione di un data quality firewall: Oracle Data Integrator assicura che i dati errati vengano automaticamente rilevati e riciclati prima dell'inserimento nell'applicazione target. Ciò viene eseguito senza la presenza di programmazione, sfruttando le regole di integrità dei dati e dei vincoli definiti sia sulla target applicatione che in Oracle Data Integrator.

- Migliori performance di esecuzione: il sovracitato cambiamento nell'architettura di ETL di Oracle Data Integrator permette migliori risultati in fase di runtime.

- Indipendenza della piattaforma: Oracle Data Integrator supporta tutte le piattaforme, software e hardware;

- Data connectivity: Oracle Data Integrator supporta tutti i RDBMS, comprese le principali piattaforme per Data Warehouse, e numerose altre tecnologie fra cui i flat file, ERP, LDAP e XML.

- Risparmio sui costi: l'eliminazione del server e del motore ETL permettono la riduzione dei costi iniziali e di quelli di mantenimento.

Capitolo 2

LA PROGETTAZIONE FUNZIONALE

Dopo aver presentato, nel precedente capitolo introduttivo, gli obiettivi che si vogliono ottenere con il presente progetto di tesi, ovvero l'analisi comparativa tra due strumenti utilizzabili per la realizzazione degli estrattori dati per l'alimentazione periodica di un data warehouse, si andranno di seguito a descrivere nel dettaglio i processi di ETL atti a popolare le tabelle di destinazione finale alimentanti i corrispondenti cubi. A titolo di completezza del confronto verrà anche simulato e considerato un caso di modifica, da parte del cliente, dei requisiti precedentemente commissionati per entrambe le estrazioni (si pensi al caso in cui venga richiesta la modifica di una delle dimensioni da considerare per l'analisi del fatto del cubo).

Le informazioni analitiche estratte riguardano principalmente il ciclo passivo di Siram e sono utilizzate dagli utilizzatori finali del cubo per trarre indicazioni sull'andamento aziendale, in maniera tale da poter, se necessario, intervenire suggerendo adeguate azioni correttive. Queste estrazioni sono effettuate periodicamente, generalmente a valle del consolidamento del dato sorgente che corrisponde alla chiusura mensile dei conti. Nel momento in cui, in seguito, dovremo definire, come parametro richiesto dai programmi in PL/SQL o dalle interfacce del software, il periodo di estrazione per cui realizzare i prospetti, sarà utilizzato come periodo campione il mese di Ottobre dell'anno 2008 (tutte le indagini, quindi, faranno riferimento a quanto

accaduto durante questo mese).

A partire da questa fase di progettazione, si procederà, nei due capitoli successivi, alla descrizione delle operazioni tecnico/implementative che hanno permesso la realizzazione delle estrazioni considerate sia con i programmi/-script in PL/SQL che con lo strumento Oracle Data Integrator: questo sarà il punto di partenza che permetterà di trarre le considerazioni di paragone tra le due soluzioni.

2.1 Dimensioni di analisi

Tra i processi aziendali di cui i dirigenti richiedono l'analisi tramite il caricamento dei cubi nel Data Warehouse e, a partire da essi, la realizzazione di rapporti sintetici (ottenuti mediante l'utilizzo degli operatori OLAP), quelli oggetto del presente progetto di tesi sono relativi all'analisi degli acquisti dei materiali dai fornitori e all'analisi delle ore lavorate su commessa cliente (preventivo/consuntivo).

Per entrambi i cubi, quindi, l'obiettivo del processo di ETL sarà quello dell'estrazione e del caricamento dei dati filtrati ed aggregati per poter alimentare la base dati su cui poggia il motore di query. I requisiti analizzati che includono le business rules considerate per l'estrazione dei dati significativi per l'indagine, sono ricavate direttamente dal documento di analisi dei requisiti, la cui stesura, effettuata in seguito all'intervista del cliente finale, è il punto di partenza della progettazione concettuale e logica dei cubi, che poi porterà alla fase di sviluppo vera e propria del processo ETL.

Per quel che riguarda il cubo chiamato "Ore Lavorate", l'esigenza da parte del cliente è quella di confrontare quantità e costi corrispettivi sia a livello di preventivo che di consuntivo delle ore lavorate rispetto a diverse dimensioni di analisi, precisamente le commesse, le risorse (o employee), il periodo temporale, il luogo di esecuzione dei lavori e aggregazione dei costi, la squadra e il contratto. Il confronto è significativo solo per le commesse di tipo operativo (commesse di servizio verso i clienti) e per tipi di spesa (o expenditure type) di un numero limitato di tipologie. E' inoltre necessario che sia implementato un ulteriore filtro relativo alle tipologie di budget utilizzati.

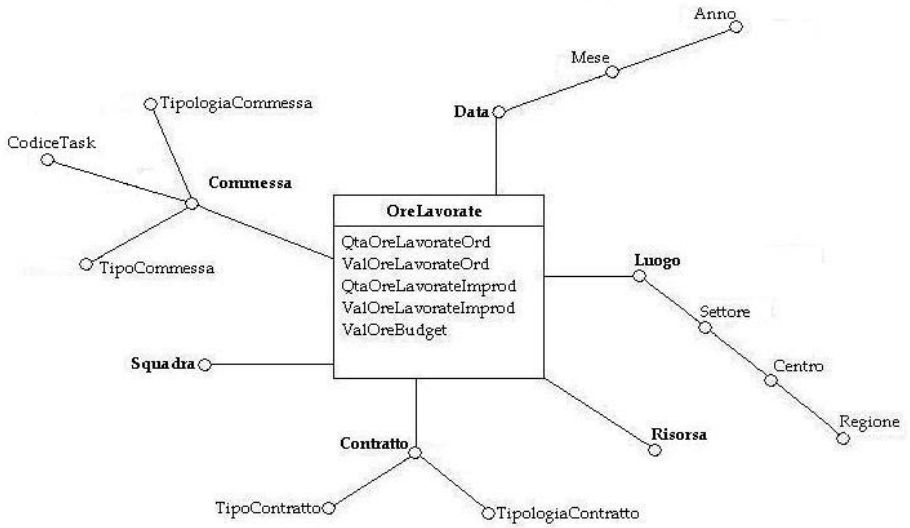


Figura 2.1: Data mart: ore lavorate

Descritti il fatto, le misure e le dimensioni da considerare per questo cubo, è possibile definire il suo schema concettuale, che viene mostrato in figura 2.1.

Per quanto riguarda invece il cubo "Ordini Materiali", in esso vengono mantenute informazioni di sintesi concernenti le spese sostenute per l'approvvigionamento delle materie prime, necessarie per l'esecuzione dei lavori commissionati dai clienti.

Questa volta i dati aggregati che dovranno comparire nei report analizzati dai dirigenti saranno i valori monetari dell'importo ordinato, fatturato e del valore del rateo dell'ordine rispetto a diverse dimensioni, fra cui il tipo di conto e categoria dell'ordine richiesto, la divisione dell'azienda in cui avvengono i lavori, il periodo temporale e il fornitore. Il cliente richiede inoltre che gli ordini da considerare nei prospetti e per i quali quindi calcolare le tre misure aggregate, descritte in precedenza, debbano avere delle determinate caratteristiche, per esempio essere già stati approvati e rientrare in alcune specifiche categorie.

Anche in questo caso, fatte tali premesse, è possibile definire lo schema concettuale del cubo, visualizzato in figura 2.2

Nei prossimi due paragrafi verrà descritto dettagliatamente il processo di

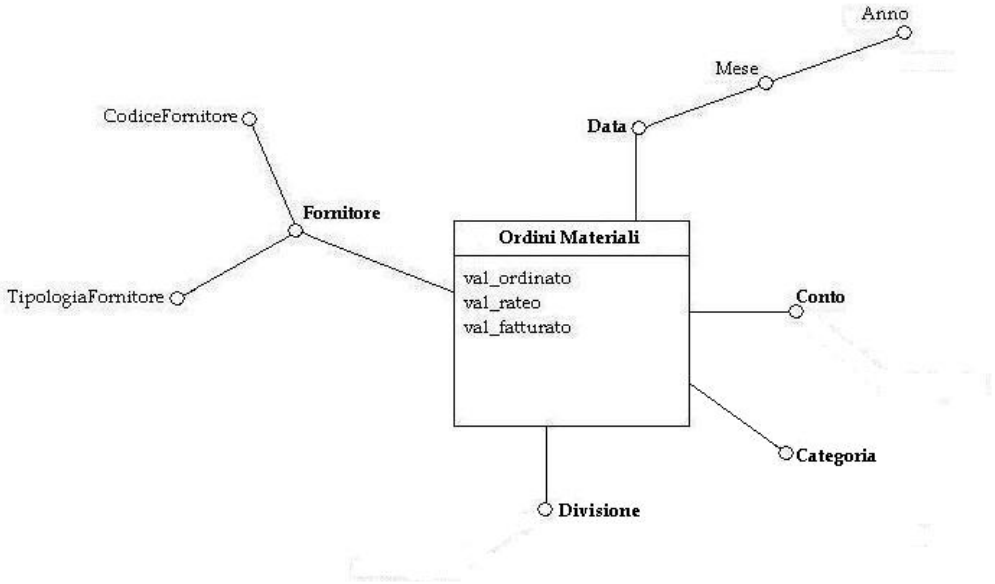


Figura 2.2: Data mart: ordini materiali

estrazione e trasformazione dei dati necessari per il caricamento del cubo. Per ognuna di esse si procederà analizzando il flusso di dati da realizzare per il loro popolamento, descrivendo le tabelle sorgenti da cui prelevare i dati, le business rules da implementare e il data mapping da effettuare per definire le sorgenti dei campi che andranno ad ospitare.

2.2 Il flusso ETL del cubo Ore Lavorate

Le tabelle che vanno realizzate per eseguire l'alimentazione del cubo delle ore lavorate sono due e vengono chiamate "DWBUDPR" e "DWPINTE". Mentre nella tabella DWPINTE vengono memorizzati i consuntivi delle ore lavorate rispetto alle dimensioni da considerare, quella chiamata DWBUDPR sintetizzerà il quantitativo di ore lavorate e il corrispettivo valore monetario previsto nel budget iniziale e messo a disposizione per ciascuna commessa in lavorazione nel periodo di estrazione richiesto dall'utente. L'obiettivo, in questo caso, è quindi quello di creare delle interrogazioni che permettano di confrontare per ognuno dei lavori commissionati dai clienti, le ore effettivamente lavorate con quelle inizialmente previste.

Per comodità si divide il flusso ETL che deve essere implementato in più step: ciò permette di isolare le fasi di pulizia e normalizzazione dei dati e illustrare in maniera più chiara il contenuto delle informazioni che si sta trattando e le regole di business da implementare.

Nel primo step si estraggono i record che il cliente ha interesse ad esaminare in una tabella intermedia, chiamata "RICERCA PROJECT", applicando i filtri che implementano i requisiti espressi dal cliente; in questo modo la fase successiva di analisi dei dati aggregati, peraltro, sarà resa più veloce dal fatto di avere meno record per cui effettuare i calcoli. Nella figura 2.3 vengono elencate le tabelle da cui si prelevano i dati per l'estrazione della suddetta tabella di appoggio, con affianco una breve descrizione per ognuna di esse.

Nome tabelle sorgenti	*Descrizione*
PA_PROJECTS_ALL	Tabella di anagrafica delle commesse
ERS_PA_ORG_HIERARCHY_V	Vista contenente altre info. sulle commesse, in particolar modo relative alla loro gerarchia organizzativa

Figura 2.3: Tabelle sorgenti: RICERCA_PROJECT

Nella figura 2.4, invece, vengono mostrati i dettagli relativi al data mapping delle colonne della tabella target.

Colonna target	*Colonna di origine*	*Implementazione*
Id_commessa	PA_PROJECTS_ALL.project_id	PA_PROJECTS_ALL.project_id
Codice_commessa	PA_PROJECTS_ALL.segment1	PA_PROJECTS_ALL.segment1
Nome_commessa	PA_PROJECTS_ALL.name	PA_PROJECTS_ALL.name
Data_inizio_commessa	PA_PROJECTS_ALL.start_date	PA_PROJECTS_ALL.start_date
Data_fine_commessa	PA_PROJECTS_ALL.completion_date	PA_PROJECTS_ALL.completion_date
Sector_raggr	ERS_PA_ORG_HIERARCHY_V.center	ERS_PA_ORG_HIERARCHY_V.center
Sector_raggr_id	ERS_PA_ORG_HIERARCHY_V.center_id	ERS_PA_ORG_HIERARCHY_V.center_id
Sector_name	ERS_PA_ORG_HIERARCHY_V.sector	ERS_PA_ORG_HIERARCHY_V.sector
Sector_id	ERS_PA_ORG_HIERARCHY_V.sector_id	ERS_PA_ORG_HIERARCHY_V.sector_id
Region_name	ERS_PA_ORG_HIERARCHY_V.region	ERS_PA_ORG_HIERARCHY_V.region
Region_id	ERS_PA_ORG_HIERARCHY_V.region_id	ERS_PA_ORG_HIERARCHY_V.region_id

Figura 2.4: Mapping: RICERCA_PROJECT

Per i record da mantenere, vengono memorizzati i valori delle dimensioni rispetto a cui raggruppare i dati aggregati, citate nel precedente paragrafo.

Dalle informazioni contenute nella figura 2.5 (tabella delle business rules da rispettare per l'estrazione dei record), si evince che le commesse per cui il cliente vuole eseguire l'indagine sono soltanto quelle operative: di tipo "Management", ovvero commesse relative a contratti di gestione, che solitamente prevedono l'erogazione dei servizi di "Attività di fornitura di combustibile" e di "Attività di manutenzione", e di tipo "Work", cioè commesse che raggruppano costi e ricavi relativi a servizi di tipo "Lavori (indotti e installazioni)"; inoltre è necessario considerare solo di quei lavori per i quali sono state effettuate delle ore lavorate nel periodo oggetto dell'estrazione, e nel caso specifico sono realizzati da Siram, dal momento che il sistema informativo supporta anche altre società del gruppo Dalkia.

Descrizione	*Implementazione*
Prendiamo in considerazione solo le commesse di tipo "work" o "management" perchè rappresentano le commesse di tipo operazionale	substr(PA_PROJECTS_ALL.segment1,1,1) in ('M', 'W')
Memorizziamo soltanto le commesse che sono in corso, rispetto alla data di estrazione indicata	p_data_estrazione between to_char(PA_PROJECTS_ALL.start_date,'YYYYMM') and to_char(nvl(PA_PROJECTS_ALL.completion_date,to_date('31/12/2999','DD/MM/YYYY')),'YYYYMM')
Memorizziamo solo le commesse di Siram (p_org_id=104)	PA_PROJECTS_ALL.org_id = p_org_id

Figura 2.5: Business rules: RICERCA_PROJECT

I dati contenuti nella tabella appena creata vengono sfruttati nel secondo step del flusso ETL per il popolamento delle due tabelle finali.

Nome tabelle sorgenti	*Descrizione*
RICERCA_PROJECT	La tabella creata in precedenza che contiene solo le commesse da considerare per il rapporto
ERS_CDG_PA_PROJECT_BUDGET_V	Una vista che contiene informazioni sul budget di tutte le commesse
PA_BUDGET_VERSIONS	Tabella che contiene le versioni dei budget approvati

Figura 2.6: Tabelle sorgenti: DWBUDPR

Si inizia ora a considerare per prima l'estrazione della tabella DWBUDPR. Nella figura 2.6 si elencano le tabelle sorgenti da cui vengono prelevate le informazioni necessarie.

Nella figura 2.7 sono illustrati i dettagli del processo di data mapping dei campi della tabella target, definendone per ognuno l'implementazione e di conseguenza le tabelle source da cui recuperare le informazioni. I dati aggregati vengono calcolati effettuando la sommatoria delle ore lavorate e definendo il loro corrispettivo valore in termini monetari, raggruppando per commessa. Completano il rapporto informazioni relative al mese ed anno di estrazione e alla data in cui la tabella è stata aggiornata (o popolata per la prima volta).

Colonna target	*Colonna di origine*	*Implementazione*
Anno_compe	#data_estrazione	substr(#data_estrazione,1, 4)
Mese_compe	#data_estrazione	substr(#data_estrazione,5,2)
Codice_commessa	RICERCA_PROJECT.CODICE_COMMESSA	RICERCA_PROJECT.CODICE_COMMESSA
Valore_budget	ERS_CDG_PA_PROJECT_BUDGET_V.BDG_RAW_COST	sum(nvl(ERS_CDG_PA_PROJECT_BUDGET_V.BDG_RAW_COST,0))
Ore_budget	ERS_CDG_PA_PROJECT_BUDGET_V.BDG_QUANTITY	sum(nvl(ERS_CDG_PA_PROJECT_BUDGET_V.BDG_QUANTITY,0))
Data_aggiornamento		SYSDATE

Figura 2.7: Mapping: DWBUDPR

Le business rules da implementare, elencate nella figura 2.8, sono soprattutto relative a restrizioni e limitazioni (filtri) sul budget che deve essere preso in considerazione per l'estrazione delle informazioni richieste. In particolare si richiede che il budget contenga informazioni sia sui costi che sui ricavi delle commesse, che sia stato approvato, che sia valido rispetto alla data di estrazione immessa e che non sia di tipo forecast (ovvero una riprevisione a lavori avviati delle commesse in questione), bensì la stima iniziale delle ore necessarie all'esecuzione del lavoro. Soltanto le ore relative ai tipi di spesa (expenditure type) "Pers. P2" o "Pers. Lavori Indo" verranno prese in considerazione per il calcolo dei risultati, cosi come richiesto nei requisiti.

Descritte tabelle sorgenti, business rules e data mapping relativi alla tabella che riassume le previsioni delle ore lavorate per le varie commesse, si

Descrizione	*Implementazione*
Teniamo solo gli expenditure i cui tipi rientrano fra quelli "pers. P2" o "pers. Lavori Indo"	ERS_CDG_PA_PROJECT_BUDGET_V.EXPENDITURE_TYPE in ('MANO D''OPERA PERS. P2', 'MANO D''OPERA PERS. LAVORI INDO', '##C','##D')
Il livello di dettaglio da considerare per il calcolo delle ore lavorate è quello dell'expenditure type	ERS_CDG_PA_PROJECT_BUDGET_V.RESOURCE_TYPE_CODE= 'EXPENDITURE_TYPE'
Il budget da considerare deve essere valido rispetto alla data di estrazione considerata	to_date(#data_estrazione||'01','YYYYMMDD') between ERS_CDG_PA_PROJECT_BUDGET_V.BDG_START_DATE and ERS_CDG_PA_PROJECT_BUDGET_V.BDG_END_DATE
Consideriamo budget e non forecast	ERS_CDG_PA_PROJECT_BUDGET_V.CLASSIFICAZIONE_BUDGET='BDG'
Consideriamo solo quei budget che contiengono info. sia sui costi che sui ricavi	PA_BUDGET_VERSIONS.VERSION_TYPE='ALL'
Il budget deve essere stato approvato	PA_BUDGET_VERSIONS.BUDGET_STATUS_CODE='B'

Figura 2.8: Business rules: DWBUDPR

passa ora ad effettuare le stesse operazioni per l'analisi dell'estrazione della tabella "DWPINTE".

L'elenco delle tabelle sorgenti è mostrato nella figura 2.9. Tramite esse sarà possibile recuperare i campi relativi alle varie dimensioni considerate, ovvero i punti di vista da cui poter in seguito osservare in modo differente le misure di analisi.

Nome tabelle sorgenti	*Descrizione*
RICERCA_PROJECT	La tabella creata in precedenza che contiene solo le commesse da considerare per il rapporto
PA_TASKS	Anagrafica dei task relativi alle commesse
PA_EXPENDITURES_ALL	Anagrafica degli expenditure relativi ai vari task
PA_EXPENDITURE_ITEMS_ALL	Anagrafica degli expenditure item che possono essere imputati per le commesse
ERS_CDG_PA_PROJECT_TASK_V	Vista che contiene altre info. di interesse relative ai task
PER_ALL_PEOPLE_F	Anagrafica degli employee responsabili dell'imputazione del costo delle ore relativamente alle varie commesse

Figura 2.9: Tabelle sorgenti: DWPINTE

Il data mapping delle colonne target è invece visualizzato nella figura 2.10.

Colonna target	*Colonna di origine*	*Implementazione*
Stagione	ERS_CDG_PA_PROJECT_TASK_V.STAGIONE	ERS_CDG_PA_PROJECT_TASK_V.STAGIONE
Anno_rapportino	#data_estrazione	substr(#data_estrazione,1,4)
Mese_rapportino	#data_estrazione	substr(#data_estrazione,5,2)
Codice_risorsa	PER_ALL_PEOPLE_F.EMPLOYEE_NUMBER	PER_ALL_PEOPLE_F.EMPLOYEE_NUMBER
Regione_commessa	RICERCA_PROJECT.REGION_NAME	RICERCA_PROJECT.REGION_NAME
Raggruppamento_CDC	RICERCA_PROJECT.SECTOR_RAGGR	RICERCA_PROJECT.SECTOR_RAGGR
Codice_CDC	RICERCA_PROJECT.SECTOR_NAME	RICERCA_PROJECT.SECTOR_NAME
Codice_commessa	RICERCA_PROJECT.CODICE_COMMESSA	RICERCA_PROJECT.CODICE_COMMESSA
Tipo_commessa		Null
Tipologia_commessa		Null
Numero_task	PA_TASKS.TASK_NUMBER	PA_TASKS.TASK_NUMBER
Codice_prestazione		Null
Codice_attivita		Null
Codice_squadra		Null
Numero_ore_ord	PA_EXPENDITURE_ITEMS_ALL.QUANTITY	nvl(PA_EXPENDITURE_ITEMS_ALL.QUANTITY,0)
Numero_ore_improd		Null
Valore_ore_ord	PA_EXPENDITURE_ITEMS_ALL.BURDEN_COST	nvl(PA_EXPENDITURE_ITEMS_ALL.BURDEN_COST,0)
Valore_ore_improd		Null
Data_aggiornamento		SYSDATE

Figura 2.10: Mapping: DWPINTE

A differenza di quanto visto in precedenza, in questo caso, per il calcolo delle ore consuntive si richiede un livello di dettaglio maggiore. Quello che viene fatto è sostanzialmente "esplodere" le informazioni della singola commessa in maniera da ottenere dettagli sui costi più precisi relativamente alle sotto-attività, o task, di cui ognuna di essa si compone. Ciò consente di impostare una granularità delle osservazioni molto fine, che permette agli utilizzatori finali del cubo di estrarre dai prospetti informazioni più precise.

Nella tabella di destinazione saranno mantenuti i campi relativi alle dimensioni rispetto alle quali poter aggregare le informazioni numeriche, ovvero ore lavorate e improduttive e corrispondenti valori monetari (che stavolta non sono quelli previsti, bensì quelli effettivamente consuntivati). Dei campi, relativi ad alcune delle dimensioni considerate, vengono creati ma non valorizzati. Tali informazioni sono oggetto di integrazione e completamento da

fonti esterne prima dell'effettivo caricamento del cubo nel data warehouse. Infine compaiono informazioni relative alla data considerata per l'estrazione e al giorno mese e anno in cui la tabella è stata aggiornata (o popolata per la prima volta).

Le restrizioni presenti in questa fase (figura 2.11) consentono di raffinare il risultato in modo tale da tener conto soltanto delle ore lavorate registrate nel mese che comprende la data di estrazione e i cui tipi di spesa associati siano di tipo "Pers. P2" o "Pers. Lavori Indo", cosi come visto anche per l'estrazione precedente.

Descrizione	*Implementazione*
Il budget da considerare deve essere valido rispetto alla data di estrazione considerata	to_date(#data_estrazione||'01','YYYYMMDD') between ERS_CDG_PA_PROJECT_BUDGET_V.BDG_START_DATE and ERS_CDG_PA_PROJECT_BUDGET_V.BDG_END_DATE
Teniamo conto solo degli expenditure che sono stati registrati nel mese di estrazione	to_char(PA_EXPENDITURE_ITEMS_ALL.EXPENDITURE_ITEM_DATE,'YYYYMM') = #data_estrazione

Figura 2.11: Business Rules: DWPINTE

Si rimanda ai prossimi capitoli per l'analisi dell'implementazione tecnica delle fasi finora trattate dal punto di vista funzionale; sarà possibile inoltre analizzare parte dei record inclusi nelle tabelle create, a titolo di esempio.

2.3 Il flusso ETL del cubo Ordini Materiali

La tabella da estrarre per il caricamento del cubo che consente l'analisi degli ordini viene chiamata "DWFATVE2". Essa ospiterà i campi aggregati e le dimensioni di analisi illustrate nel paragrafo iniziale di questo capitolo.

Anche in questo caso il processo ETL è scomposto in più passi e sono utilizzate delle tabelle di appoggio per gli stessi motivi citati in precedenza. Inizialmente viene creato l'archivio che memorizza solo gli ordini indicati nel documento dei requisiti, per i quali si vuole svolgere l'indagine; questo inoltre consente di alleggerire il carico computazionale necessario in seguito per il calcolo degli aggregati. Già in questa fase, quindi, vengono implementate

alcune delle business rules necessarie per soddisfare le necessità del cliente. Nella figura 2.12 sono elencate le tabelle sorgenti, ospitate nel sistema informativo del gruppo Dalkia, da cui prelevare i dati necessari.

Nome tabelle sorgenti	*Descrizione*
PO_HEADERS_ALL	Anagrafica degli ordini richiesti ai fornitori
PO_LINES_ALL	Anagrafica delle linee (1 ordine ha N linee) riferite agli ordini
PO_LINES_LOCATIONS_ALL	Anagrafica degli shipments (1 linea ha N shipments) riferiti alle linee
PO_DISTRIBUTIONS_ALL	Anagrafica delle distributions (1 shipments ha N distributions) riferite agli shipments
PO_VENDORS_ALL	Anagrafica dei fornitori delle aziende del gruppo Dalkia
PO_VENDOR_SITES_ALL	Anagrafica delle sedi dei fornitori delle aziende del gruppo Dalkia
MTL_CATEGORIES_B	Tabella che memorizza informazioni sulla classificazione degli acquisti

Figura 2.12: Tabelle sorgenti: RICERCA_VENDOR_CONTO

In particolare, fra queste, hanno una certa rilevanza le quattro tabelle, ciascuna in rapporto uno a molti con l'altra, che sono adibite alla memorizzazione dei dettagli riguardanti gli ordini richiesti dalle società appartenenti al gruppo Dalkia. Per un ordine richiesto da una determinata azienda devono essere memorizzate le linee di cui si compone, ovvero l'elenco degli articoli (se sono più di uno) richiesti al fornitore per la spedizione. Inoltre a ogni linea possono corrispondere più spedizioni o consegne, ovvero le disposizioni di invio dei materiali verso le destinazioni finali. Infine a una spedizione/consegna possono corrispondere più "distributions", le quali servono per imputare la quota del costo relativo alla commessa che ha richiesto quel tipo di materiale.

E' necessario inoltre, per comprendere le informazioni presenti nei campi relativi, fare un accenno al contenuto della tabella MTL_CATEGORIES_B, che tiene traccia della categoria dell'acquisto effettuato, memorizzato in quattro livelli gerarchici (i quattro segment). Dato l'acquisto di un determinato articolo, questi servono a specificare sempre più dettagliatamente in che categoria di acquisto esso rientra.

In figura 2.13 viene mostrato il data mapping relativo alle colonne della tabella di appoggio considerata. Nei suoi campi sarà allocato spazio per la memorizzazione delle dimensioni definite in precedenza, ovvero il fornitore,

a cui si richiede il materiale, il tipo di acquisto insieme alla sua categoria (e ai quattro segmenti di cui si compone) e alla divisione aziendale che lo richiede. L'informazione sul conto dell'acquisto, ultima dimensione di analisi, viene invece al momento mantenuta priva di valori, e sarà popolata nella fasi finali dell'estrazione.

Colonna target	*Colonna di origine*	*Implementazione*
Vendor_id	PO_HEADERS_ALL	PO_HEADERS_ALL VENDOR_ID
Codice_fornitore	PO_HEADERS_ALL	PO_VENDORS.SEGMENT1
Divisione_fornitore	PO_VENDOR_SITES_ALL	substr(nvl(PO_VENDOR_SITES_ALL.ATTRIBUTE4,'??'),1,2)
Expenditure Type	PO_VENDORS	PO_VENDORS.EXPENDITURE_TYPE
Categoria	MTL_CATEGORIES_B.CATEGORY_ID	MTL_CATEGORIES_B.CATEGORY_ID
Segment1_category	MTL_CATEGORIES_B.SEGMENT1	substr(nvl(MTL_CATEGORIES_B.SEGMENT1,'00'),1,2)
Segment2_category	MTL_CATEGORIES_B.SEGMENT2	substr(nvl(MTL_CATEGORIES_B.SEGMENT2,'00'),1,2)
Segment3_category	MTL_CATEGORIES_B.SEGMENT3	substr(nvl(MTL_CATEGORIES_B.SEGMENT3,'00'),1,2)
Segment4_category	MTL_CATEGORIES_B.SEGMENT4	substr(nvl(MTL_CATEGORIES_B.SEGMENT4,'00'),1,2)
Conto		Null

Figura 2.13: Mapping: RICERCA_VENDOR_CONTO

Infine in figura 2.14 sono mostrate le business rules che permettono di restringere, in base alle richieste, il numero di record da estrarre per le successive fasi di implementazione.

Descrizione	*Implementazione*
L'ordine deve essere stato approvato durante il periodo di estrazione considerato	trunc(PO_HEADERS_ALL.approved_date) between p_data_estr_iniz and p_data_estr_fine
L'ordine deve essere di tipo standard	PO_HEADERS_ALL.TYPE_LOOKUP_CODE='STANDARD'
Deve essere definitio il tipo di spesa dell'ordine considerato	PO_VENDORS.EXPENDITURE_TYPE is not null
L'ordine deve essere stato richiesto dalla società Siram (ord_id=104)	PO_HEADERS_ALL.ORG_ID=#org_id
L'ordine deve essere stato approvato	PO_HEADERS_ALL.AUTHORIZATION_STATUS='APPROVED'
Le quantità di pezzi acquistati per ogni articolo dell'ordine, al netto delle quantità rifiutate, devono essere maggiori di zero	(PO_LINE_LOCATIONS_ALL.QUANTITY - PO_LINE_LOCATIONS_ALL.QUANTITY_CANCELLED) > 0

Figura 2.14: Business Rules: RICERCA_VENDOR_CONTO

Affinchè i record possano essere migrati nella tabella, gli ordini cui si riferiscono devono essere relativi al periodo considerato per l'estrazione, di tipo standard ed in stato "approvato". Inoltre il tipo di spesa ad essi associato deve essere popolato da un valore fra quelli permessi, e non lasciato nullo. L'archivio appena creato viene letto nello step successivo per il calcolo dei risultati aggregati. In esso si prevede la creazione di tre tabelle temporanee, ciascuna adibita al calcolo dei tre campi precedentemente citati, ovvero importo_ordinato, importo_fatturato e importo_rateo; ciò è reso necessario dal diverso tipo di business rules di cui tener conto per il calcolo dei valori. E' implicito che per tutti gli ordini considerati in questo secondo step del processo ETL sia sempre valorizzato il campo relativo all'importo ordinato, mentre possano risultare vuoti in alcuni casi quelli relativi ai valori degli importi fatturati e dei ratei.

Al solito, si procede visualizzando tabelle sorgenti, data mapping e business rules anche per il popolamento di queste tabelle intermedie. In realtà le situazioni riguardanti le tabelle Ordinato, Fatturato e Rateo sono molto simili tra loro, a parte la modifica di poche restrizioni che comportano il collegamento talvolta con tabelle diverse per il calcolo degli aggregati. Di conseguenza andremo ad analizzare in seguito solamente l'implementazione della tabella Ordinato, citando solamente le modifiche che serve apportare per l'esecuzione delle stesse operazioni per le altre due tabelle. In figura 2.15 si visualizza l'elenco delle tabelle sorgenti necessarie per l'estrazione dei dati opportuni.

Nome tabelle sorgenti	*Descrizione*
RICERCA_VENDOR_CONTO	La tabella creata in precedenza che contiene solo i venditori e i tipi di acquisto ad essi richiesto da considerare per il rapporto
PO_HEADERS_ALL	Anagrafica degli ordini richiesti ai fornitori
PO_LINES_ALL	Anagrafica delle linee (1 ordine ha N linee) riferite agli ordini
PO_DISTRIBUTIONS_ALL	Anagrafica delle distributions (1 shipments ha N distributions) riferite agli shipments
PO_LINE_LOCATIONS_ALL	Anagrafica degli shipments (1 linea ha N shipments) riferiti alle linee
PO_VENDOR_SITES_ALL	Anagrafica delle sedi dei fornitori delle aziende del gruppo Dalkia

Figura 2.15: Tabelle sorgenti: Ordinato

Colonna target	*Colonna di origine*	*Implementazione*
V_qta_ordinato	PO_LINE_LOCATIONS_ALL .quantity	sum(PO_LINE_LOCATIONS_ALL.q uantity - PO_LINE_LOCATIONS_ALL.quanti ty_cancelled - PO_LINE_LOCATIONS_ALL.quanti ty_rejected)
V_val_ordinato	PO_LINE_LOCATIONS_ALL .quantity	sum((PO_LINE_LOCATIONS_ALL. quantity - PO_LINE_LOCATIONS_ALL.quanti ty_cancelled - PO_LINE_LOCATIONS_ALL.quanti ty_rejected)* PO_LINES_ALL.unit_price)
Vendor_id	RICERCA_VENDOR_CONT O.VENDOR_ID	RICERCA_VENDOR_CONTO.VEN DOR_ID
Expenditure_type	RICERCA_VENDOR_CONT O.EXPENDITURE_TYPE	RICERCA_VENDOR_CONTO.EXPE NDITURE_TYPE
Divisione_fornitor e	RICERCA_VENDOR_CONT O.DIVISIONE_FORNITORE	RICERCA_VENDOR_CONTO.DIVI SIONE_FORNITORE
Categoria	RICERCA_VENDOR_CONT O.CATEGORIA	RICERCA_VENDOR_CONTO.CAT EGORIA

Figura 2.16: Mapping: Ordinato

In figura 2.16 troviamo invece il data mapping delle colonne della tabella target. Quest'ultima, conterrà le dimensioni rispetto a cui i dati vengono raggruppati, e i due aggregati di interesse: la quantità dell'ordinato (valore che al momento non viene poi migrato nella tabella finale, ma comunque mantenuto in questa tabella intermedia) e il corrispettivo valore monetario, ottenuto moltiplicando ogni quantità ordinata per il corrispondente prezzo unitario.

Infine in figura 2.17 sono elencate le business rules da implementare.

Descrizione	*Implementazione*
L'ordine, deve essere stato approvato	PO_HEADERS_ALL.AUTHORIZATION_ST AT US='APPROVED'
L'ordine deve essere stato richiesto e approvato nel periodo considerato per l'estrazione	Trunc(PO_HEADERS_ALL.approved_date) between to_date('05/05/2006','DD/MM/YYYY') and to_date('27/05/2008','DD/MM/YYYY')
L'ordine deve essere stato richiesto da Siram	PO_HEADERS_ALL.ORG_ID=#org_id
L'ordine deve essere di tipo standard	PO_HEADERS_ALL.TYPE_LOOKUP_CODE='S TANDARD'

Figura 2.17: Business rules: Ordinato

Per quanto riguarda l'ordinato, è necessario estrarre i record degli ordini

di Siram, approvati, di tipo standard e con data di approvazione compresa nel periodo di estrazione considerato. Nel caso dell'estrazione della tabella Rateo, i record da estrarre sono relativi a quegli ordini per cui i materiali sono già stati ricevuti dall'azienda che li ha richiesti, ma non è ancora stata emessa una fattura. Di conseguenza, il pagamento sarà effettuato nei mesi successivi, mentre il costo sostenuto è di competenza economica del mese che si sta considerando. Infine per la realizzazione della tabella Fatturato, vanno considerati quei record per cui è stata già ricevuta la fattura da parte del fornitore.

Terminato il popolamento di tutte queste tabelle sarà ora possibile, realizzando dei join fra esse, effettuare il caricamento della tabella "DWFATVE2", nel terzo step del flusso ETL, come è possibile notare dalla figura 2.18 che elenca le tabelle sorgenti.

Nome tabelle sorgenti	*Descrizione*
RICERCA_VENDOR_CONTO	La tabella creata in precedenza che contiene solo i venditori e i tipi di acquisto ad essi richiesto da considerare per il rapporto
Ordinato	La tabella creata in precedenza che contiene il totale dell'importo ordinato per ogni forinitore e tipo di spesa considerata
Fatturato	La tabella creata in precedenza che contiene il totale dell'importo fatturato per ogni fornitore e tipo di spesa considerata
Rateo	La tabella creata in precedenza che contiene il totale dell'importo rateo per ogni fornitore e tipo di spesa considerata
PA_EXPENDITURE_TYPES	Tabella che elenca i tipi di spesa che possono essere sostenuti

Figura 2.18: Tabelle sorgenti: DWFATVE2

Sarà necessario, però, eseguire la lettura dei dati sorgenti anche dalla tabella PA_EXPENDITURE_TYPES, da cui vengono prelevate le informazioni utili per il popolamento del campo relativo al conto dell'acquisto, precedentemente non valorizzato, e trattare nuovamente il campo del rateo, a seconda di ciò che compare, in relazione allo stesso ordine, nel valore del fatturato. Se quest'ultimo sarà diverso da zero, allora esso andrà sottratto al valore del rateo precedentemente calcolato. Il risultato di questa differenza valorizzerà definitivamente il campo relativo al rateo dell'ordine in questione (in caso di numero negativo il valore risultante sarà zero).

Colonna target	*Colonna di origine*	*Implementazione*
Societa		#org_id
Divisione	RICERCA_VENDOR_CONTO.DIVISIONE_FORNITORE	RICERCA_VENDOR_CONTO.DIVISIONE_FORNITORE
Categoria		cursore.segment1_category||cursore.segment2_category||cursore.segment3_category||cursore.segment4_category
Conto	PA_EXPENDITURE_TYPES.DESCRIPTION	(rtrim(substr(PA_EXPENDITURE_TYPES.DESCRIPTION,1,instr(PA_EXPENDITURE_TYPES.DESCRIPTION,' ')-1)))
Codice_fornitore	RICERCA_VENDOR_CONTO.CODICE_FORNITORE	RICERCA_VENDOR_CONTO.CODICE_FORNITORE
Forn_convenzionato		'#'
Anno_riferimento		substr(#data_estrazione,1, 4)
Mese_riferimento		substr(#data_estrazione,5, 2)
Importo_fatturato	FATTURATO.V_VAL_FATTURATO	FATTURATO.V_VAL_FATTURATO
Importo_ordinato	ORDINATO.V_VAL_ORDINATO	ORDINATO.V_VAL_ORDINATO
Importo_rateo	RATEO.V_VAL_RATEO	RATEO.V_VAL_RATEO

Figura 2.19: Mapping: DWFATVE2

La tabella finale (data mapping delle sue colonne in figura 2.19) conterrà i campi delle quattro dimensioni, che hanno permesso inoltre il collegamento tra le tre tabelle popolate in precedenza, rispetto a cui abbiamo raggruppato finora i dati soltanto per quei record che sono stati inizialmente filtrati mediante l'estrazione della tabella RICERCA_VENDOR_CONTO. In questo archivio vengono ovviamente memorizzati anche i tre aggregati importo_ordinato, importo_fatturato e importo_rateo. Completano la tabella informazioni riguardo il periodo di estrazione considerato e la società (Siram) cui si riferisce l'analisi degli acquisti.

In questo caso non vengono mostrate informazioni relative alle restrizioni da implementare per il mantenimento dei soli record utili poichè tutte le business rules richieste fra i requisiti sono già state implementate nelle fasi precedenti.

Anche per questo processo di ETL, si rimanda ai prossimi capitoli per la sua implementazione tecnica; sarà possibile inoltre esaminare parte dei record inclusi nelle tabelle create, a titolo di esempio.

2.4 Un caso di modifica dei requisiti

Si accenna brevemente, a completamento di questo capitolo sulla progettazione funzionale dei cubi, a un ulteriore caso cui si terrà conto in seguito, in maniera tale da avere un ulteriore termine di paragone fra le due soluzioni implementate, ovvero quello della modifica dei requisiti richiesti per il caricamento del cubo, nel momento in cui l'estrattore implementato dovrà essere modificato per recepire un nuovo requisito utente.

In particolar modo si simulerà che il cliente richieda, relativamente al cubo delle ore lavorate, di poter analizzare i dati aggregati rispetto ad una nuova dimensione, ad esempio la mansione degli impiegati che hanno effettuato il lavoro. Per quanto riguarda invece il cubo dei fornitori, si terrà conto che il cliente, ad un certo punto, richieda l'analisi della situazione anche in relazione alla città di appartenenza dei venditori con cui l'azienda si relaziona.

Ciò comporta delle lievi modifiche a livello funzionale. La situazione più semplice riguarda la tabella che servirà a caricare il cubo degli acquisti, per la quale la modifica dei requisiti comporta soltanto la memorizzazione di un campo addizionale chiamato Citta_fornitore nella tabella intermedia e in quella finale. L'aggiunta di un nuovo campo non modifica in questo caso la cardinalità delle tabelle, per cui non sarà necessario imporre nuove restrizioni al flusso per ottenere il risultato desiderato.

Più consistenti risultano invece le modifiche da effettuare nel caso del cambiamento dei requisiti per il cubo delle ore lavorate. Infatti le informazioni necessarie per ricavare la mansione della risorsa non sono già disponibili nelle tabelle sorgenti del flusso ETL atto alla sua realizzazione. In particolare, i dati che ci occorrono sono contenuti nell'archivio PER_JOB_DEFINITIONS che può essere raggiunto sfruttando un campo di join di PER_ALL_PEOPLE_F ed effettuando prima il collegamento con altre due tabelle. Ciò ci consente di recuperare per ogni record riferito ad un "employee" che ha consuntivato l'ora lavorata, la mansione dello stesso all'interno dell'azienda. Se non imponessimo però delle restrizioni, potrebbe accadere di recuperare per ogni record riferito ad una risorsa, più righe riferite alla sua mansione. Questo può accadere in relazione al modello dati di gestione dei dipendenti dove ogni modifica

relativa ad un dipendente porta all'aggiunta di un nuovo record; i record precedenti vengono mantenuti ma etichettati come invalidi tramite il mantenimento di due date che informano sulla data di inizio e di fine della validità di ogni tupla. Dunque, per gestire questa situazione, per ogni tabella che colleghiamo è necessario imporre che la data odierna sia compresa nell'intervallo citato, in maniera da selezionare ogni volta soltanto l'unico record, tra i molti che si riferiscono ad un solo dipendente, attualmente valido. Ciò permette di ricavare la mansione dell'employee e inserire questa informazione in un nuovo campo della tabella DWFATVE2, evitando di modificare la cardinalità della stessa.

Capitolo 3

ETL CON ORACLE DATA INTEGRATOR

Nei precedenti capitoli si è parlato della possibilità di effettuare l'implementazione dei flussi per il popolamento delle tabelle che serviranno in seguito a caricare i due cubi tramite programmi scritti in linguaggio PL/SQL oppure interfacce sviluppate con Oracle Data Integrator.

Viene trattato di seguito il secondo tipo di soluzione proposta, definita ipotesi "Buy", perchè prevede l'acquisto di un tool appositamente pensato per scopi di ETL. E' stato dedicato in precedenza un paragrafo per la descrizione generale di questo strumento, definendo la sua architettura, i moduli grafici che lo costituiscono e i tipi di archivi che utilizza.

Si procederà adesso innanzitutto descrivendo gli step necessari per iniziare ad utilizzare il tool, e di seguito mostrando il funzionamento dei quattro moduli grafici di cui si compone. Nei successivi paragrafi sarà poi trattato lo sviluppo delle interfacce necessarie per l'estrazione delle tabelle e le modifiche da effettuare alle stesse affinchè vengano resi operativi i cambiamenti ai requisiti descritti nei paragrafi precedenti.

3.1 Gli archivi di lavoro e i moduli grafici

Per poter eseguire l'accesso a uno qualsiasi dei moduli grafici di Oracle Data Integrator, è anzitutto necessario disporre delle coordinate di connessione

Listato 3.1: Creazione dei due utenti e dei loro schemi

```
1 create user master_rep identified by welcome1 default tablespace
      users temporary tablespace temp;
2
3 grant connect, resource to master_rep;
4
5
6 create user work_rep identified by welcome1 default tablespace users
       temporary tablespace temp;
7
8 grant connect, resource to work_rep;
```

per l'accesso a un Master Repository e a un Work Repository (il motivo della necessità di connessione a questi due archivi e i dati che ognuno di essi contengono sono già stati descritti nel capitolo introduttivo). Ovviamente sarebbe stato possibile collegarsi ai repository creati in precedenza in azienda per lo sviluppo di altre interfacce, ma per completezza di esposizione simuleremo di doverne creare dei nuovi; essi saranno memorizzati in un database Oracle che si suppone per comodità già esistente.

I passi necessari per effettuare il primo accesso ai moduli grafici sono:

1. **Creazione dei due utenti che manterranno nello spazio ad essi assegnato le tabelle dei due repository**

   Ciò richiede la creazione di due schemi nel database, uno per il Master Repository e uno per il Work Repository. Nel listato 3.1 sono mostrati i comandi SQL necessari per effettuare questo step.

2. **Creazione del Master Repository**

   E' possibile creare il Master Repository tramite l'utilizzo dello strumento "Master Repository Creation", disponibile con l'installazione del software. In figura 3.1 viene visualizzato lo screenshot che mostra l'interfaccia grafica presentata al momento della sua apertura.

   All'interno della form che appare, è necessario inserire i dettagli relativi ai driver utilizzati per la connessione, l'URL per accedere al database, la sua tecnologia, user e password per l'accesso, nonchè l'id dell'archivio. Fatto ciò gli oggetti necessari vengono creati all'interno dello schema indicato.

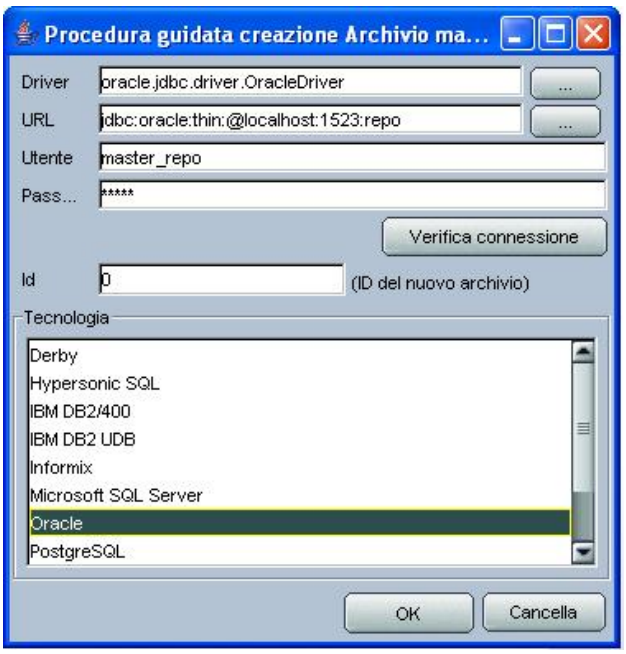


Figura 3.1: Creazione del Master Repository

3. **Connessione al Master Repository**

   Dopo aver creato il Master Repository, sarà possibile effettuare la connessione ad esso collegandosi al modulo grafico Topology Manager, che richiederà l'immissione delle coordinate di accesso all'archivio appena creato. In figura 3.2 vengono mostrate le form visualizzate per la richiesta di una nuova connessione.

   E' necessario riempire i seguenti campi:

   - Oracle Data Integrator Connection
     - Nome di accesso: Un alias generico
     - Utente: SUPERVISOR
     - Password: SUNOPSIS
   - DBMS Connection (Master Repository)
     - Utente: nome utente del proprietario delle tabelle create per il Master Repository
     - Password: password dell'utente

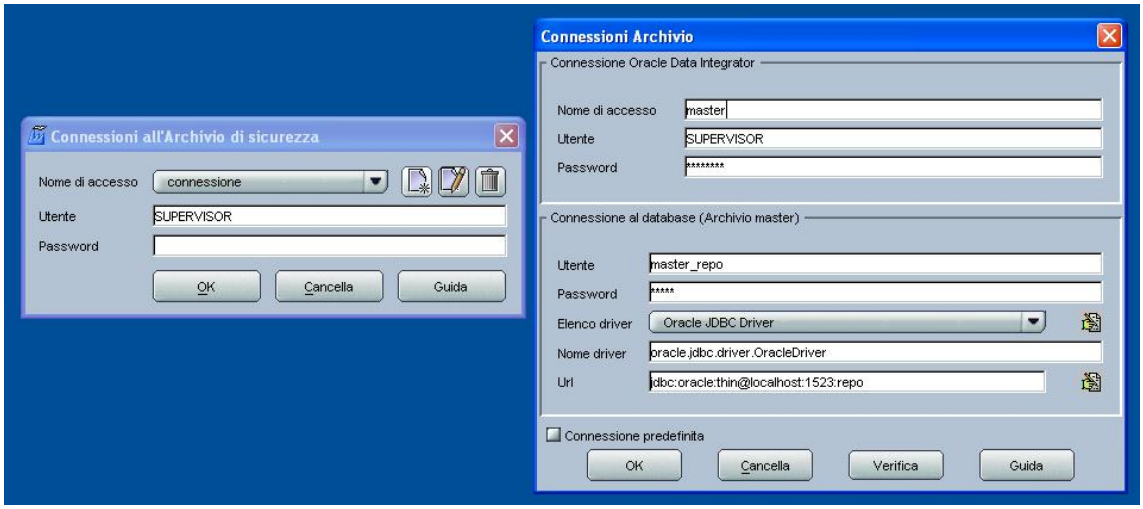


Figura 3.2: Connessione al Master Repository

- Driver: driver necessari per la connessione al db
- URL: il percorso completo del data server ospitante il repository

Fatto ciò, cliccando su OK, eseguiremo il primo accesso al modulo grafico Topology Manager.

4. **Creazione del Work Repository**

   Al suo interno si trova il pulsante che consente di impostare una nuova connessione per il Work Repository. In figura 3.3 viene visualizzata la form da compilare.

   Dovranno essere inserite le seguenti informazioni:

   - Nome: nome della connessione al Work Repository
   - Tecnologia: tecnologia del server che ospita il Work Repository
   - Utente: nome utente del proprietario delle tabelle create per il Work Repository
   - Password: password dell'utente
   - Driver: (nella scheda a fianco) driver e URL per l'accesso al db che ospita l'archivio

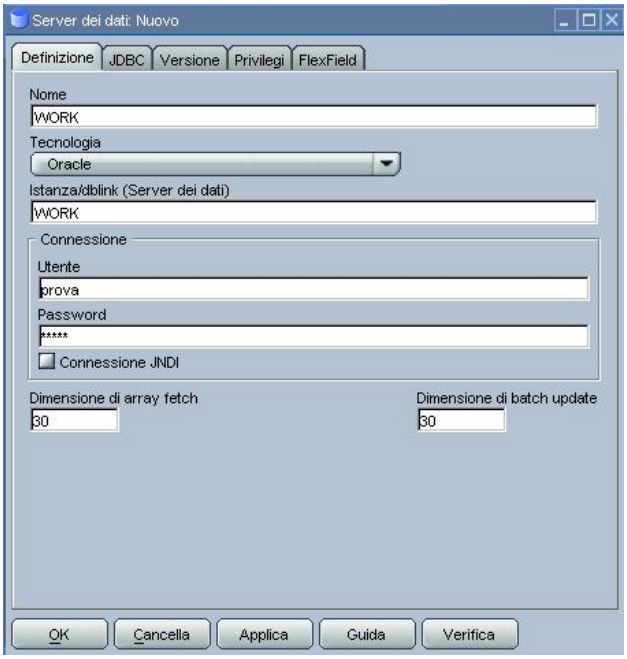


Figura 3.3: Creazione del Work Repository

Una volta riempiti i campi e dato conferma, dopo aver inserito pochi altri dettagli di connessione, il Work Repository verrà finalmente creato.

5. **Connessione al Work Repository**

   Per effettuare la connessione al Work Repository è necessario lanciare il modulo grafico Designer, che richiederà all'avvio le ultime informazioni necessarie per poter finalmente utilizzare il tool. Dettagli sono mostrati in figura 3.4.

   Di seguito di elencano i dati necessari per la connessione al Work Repository:

   - Oracle Data Integrator Connection
     - Nome di accesso: Un alias generico
     - Utente: SUPERVISOR
     - Password: SUNOPSIS
   - DBMS Connection (Master Repository)

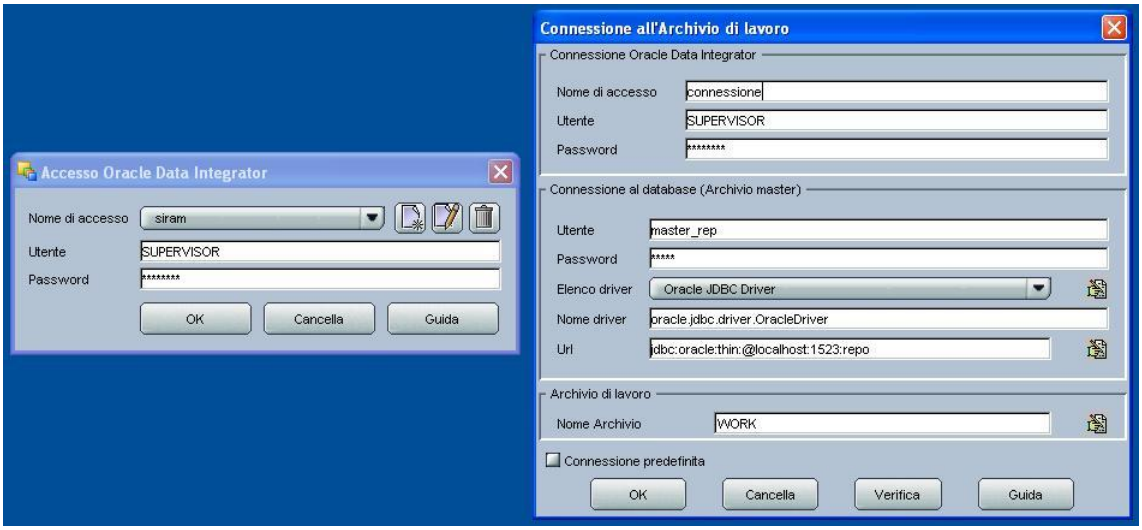


Figura 3.4: Connessione al Work Repository

  - Utente: nome utente del proprietario delle tabelle create per il Master Repository
  - Password: password dell'utente
  - Driver: driver necessari per la connessione al db
  - URL: il percorso completo del data server ospitante il repository
- Work Repository
  - Nome del Work Repository: nome assegnato all'archivio di lavoro nello step precedente

Questo completa la procedura necessaria per la prima configurazione di accesso ad Oracle Data Integrator. Prima di descrivere un altra fase importante, precedente alla creazione delle interfacce, ovvero la configurazione dell'architettura fisica e logica dei data server coinvolti nelle migrazioni, vengono qui presentati brevemente i quattro moduli grafici, in ordine indicativo di utilizzo nello svolgimento di un progetto, le cui funzionalità erano state appena accennate nella parte del primo capitolo dedicata alla trattazione di Oracle data Integrator.

Il primo strumento che viene utilizzato è proprio Topology Manager. In figura 3.5 troviamo uno screenshot che mostra una sua tipica esecuzione.

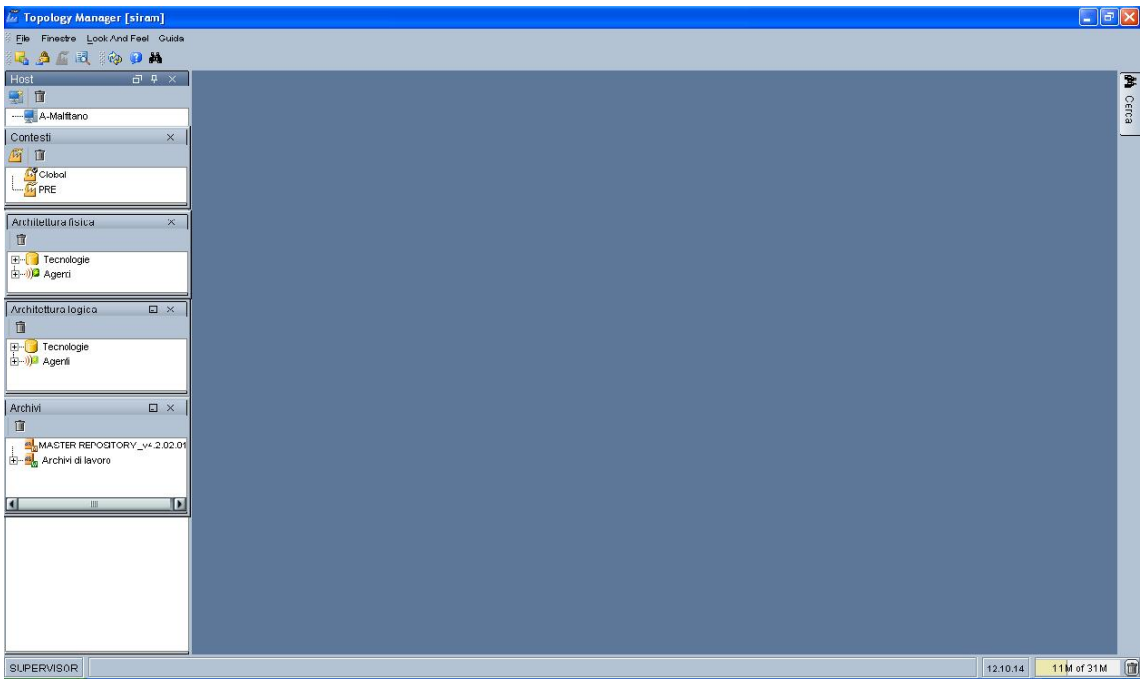


Figura 3.5: Il Topology Manager

Esso è di fondamentale importanza per la realizzazione dei progetti poichè permette di gestirne la topologia e in particolare l'architettura dei flussi delle interfacce che verranno definite attraverso il Designer. Tramite questo modulo è quindi possibile la gestione degli utenti che hanno accesso alle varie connessioni, dei contesti, ovvero dei riferimenti che permettono di collegare l'esecuzione dell'interfacce all'architettura definita per esse, dell'architettura fisica, cioè l'elenco dei data server cui collegarsi per avere accesso alle tabelle sorgenti o di destinazione dei flussi ETL, dell'architettura logica, ovvero gli schemi logici, che faranno riferimento agli schemi fisici corrispondenti, e degli archivi di lavoro che sono stati precedentemente creati e a cui si è attualmente connessi.

Il secondo strumento che dovrebbe essere utilizzato seguendo l'ordine temporale è il Security Manager, attraverso cui è possibile impostare le politiche di sicurezza necessarie per proteggere i dati contenuti all'interno dei progetti. La figura 3.6 mostra una sua tipica esecuzione.

Il Security Manager permette la creazione di utenti e profili e l'assegna-

Figura 3.6: Il Security Manager

zione dei privilegi ad essi per l'esecuzione di operazioni (edit, delete, ecc.) su determinati oggetti (data server, data types, ecc.) in determinate istanze di esecuzione (data server 1, data server 2, ecc.)

Il terzo modulo che viene preso in considerazione è il Designer, che può ipoteticamente essere considerato centrale rispetto agli altri, poichè ospita tutte le fasi relative allo sviluppo dei progetti. La figura 3.7 ne mostra una tipica esecuzione.

Nel Designer possono essere ritenute di particolare importanza due schede: quella relativa ai progetti e quella relativa ai modelli. Tramite la prima è possibile la creazione di un nuovo lavoro, o di uno qualsiasi degli elementi che possono costituirlo, dalle interfacce alle procedure, dalle variabili alle sequenze. Particolare rilevanza assumono proprio le interfacce, che permettono di implementare un flusso dati di ETL e che serviranno nei prossimi paragrafi per effettuare l'estrazione dei dati. Un accenno meritano anche i knowledge modules, che hanno una rilevanza decisiva all'interno dello sviluppo delle interfacce perchè rappresentano il modo tramite cui ODI permette facilmente

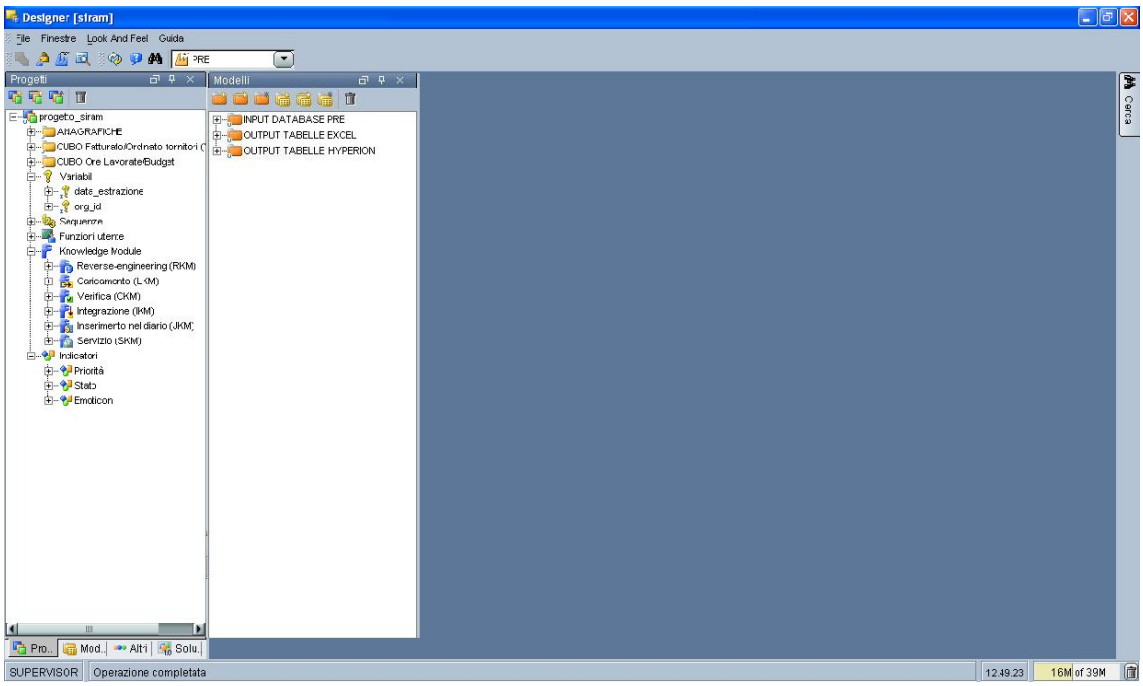


Figura 3.7: Il Designer

la connessione con la maggior parte delle tecnologie disponibili di server dati, una funzionalità fondamentale per un software di integrazione che pretenda di essere leader nel suo settore di mercato. Volendo anticipare un raffronto con la programmazione tradizionale sul modo in cui viene realizzata la connessione alle sorgenti informative, si potrebbe dire che i knowledge module risparmiano diverse linee di codice PL/SQL che dovrebbero essere scritte per realizzare questo scopo. I knowledge modules vengono classificati in base al tipo di operazioni che permettono di svolgere. In particolar modo i KM che vedremo nel proseguo della trattazione sono quelli di caricamento (LKM), verifica (CKM) e integrazione (IKM). Per ognuna di queste categorie, sono implementati più knowledge modules, che si distinguono proprio per i tipi di data server da cui possono leggere i dati, controllarli o in cui caricarli: esistono ad esempio knowledge modules che permettono la lettura dei dati da database Oracle, SQL Server, da file di testo, file csv ecc. Nel nostro caso sarebbe opportuno selezionare knowledge module che permettano le lettura dei dati sorgenti da database Oracle (poichè da esso recupereremo i dati di par-

tenza per i flussi) e il caricamento di quelli trasformati in Hyperion Essbase (laddove dovranno essere create le tabelle di destinazione). Ogni knowledge module definisce le sequenze di comandi SQL che devono essere lanciati per realizzare le tre fasi che compongono l'estrazione dei dati. Ovviamente sarà permessso modificare le operazioni che compongono un determinato knowledge module, in maniera da poter personalizzare quanto più possibile il proprio flusso ETL.

L'altra scheda importante presente all'interno del Designer è quella relativa ai modelli. Le interfacce, affinchè possano essere lanciate, necessitano l'inserimento di una o più tabelle da cui leggere i dati e di una in cui caricare quelli trasformati. Affinchè queste tabelle siano disponibili per essere considerate durante l'esecuzione di un flusso, è necessario che esse siano incluse in dei modelli. Per far ciò si dovrà innanzitutto inserire il nome dello schema dell'utente che possiede le tabelle considerate, e in seguito effettuare il reverse engineering di quelle il cui contenuto si vuole far mantenere nel modello creato. Questi passaggi saranno trattati dettagliatamente nel paragrafo dedicato allo sviluppo delle interfacce, nel momento in cui sarà necessario leggere le tabelle che ospitano i dati necessari per far partire il flusso.

Infine l'ultimo strumento utilizzato in ordine temporale per l'esecuzione del primo progetto dovrebbe essere l'Operator. Esso consente di seguire l'andamento del lancio di un'interfaccia, sessione o scenario, in maniera tale da poter identificare eventuali errori occorsi durante l'esecuzione delle varie operazioni. Sarà possibile, come vedremo in seguito, seguire ogni singolo passo che verrà effettuato durante il flusso di ETL e identificare l'insieme di comandi SQL eseguiti. In caso di problemi, l'Operator provvederà a segnalarli mediante un apposito segnale di avviso e restituirà il codice del problema che si è verificato. Verrà mostrata una schermata di esecuzione dell'Operator poche pagine più avanti, nel momento in cui tramite esso potremo osservare l'andamento del lancio della prima interfaccia.

3.2 La creazione della topologia del progetto

Tramite il modulo grafico Topology Manager, l'utente può creare una vera e propria cartografia fisica e logica delle architetture e dei componenti che sono

necessari per l'implementazione dei flussi di ETL. Anche per lo sviluppo del nostro progetto di caricamento dei due cubi, è stata necessaria la realizzazione di una topologia, e adesso saranno mostrati gli step che sono stati seguiti per realizzare queste operazioni, di fondamentale importanza, poichè senza il loro completamento lo sviluppo delle interfacce non potrebbe mai avere luogo:

- **Creazione del contesto di esecuzione**

  Per la creazione del contesto di esecuzione (la cui utilità sarà probabilmente più chiara nel momento in cui si tratterà lo sviluppo delle interfacce) è necessario riempire i campi della finestra di inserimento nuovo contesto, mostrati in figura 3.8.

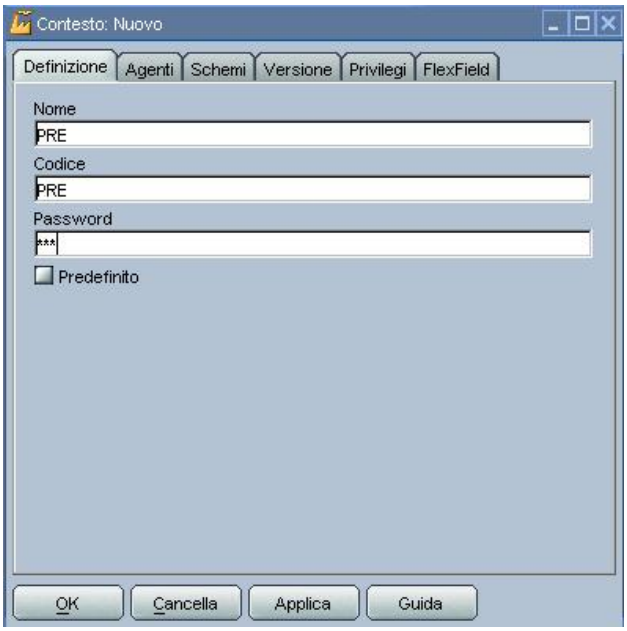


Figura 3.8: Creazione nuovo contesto

Essi sono:

  - Nome: nome che servirà per richiamare il contesto di esecuzione
  - Codice: riempito automaticamente da ODI
  - Password: password che verrà richiesta ogni volta che sarà richiamato il contesto

- **Creazione dei data server da utilizzare nelle fasi di sviluppo**

  Un data server corrisponde a un database che deve essere connesso ad Oracle Data Integrator, in maniera tale che i moduli grafici possano interagire, tramite comandi SQL, con esso. Sarà necessaria la creazione di un data server ogni qualvolta dovremo essere in grado di leggere dati da tabelle contenute in database non ancora connessi con ODI. La scheda da compilare per effettuare questa operazione, cui si accede scegliendo prima la tecnologia delle tabelle da leggere, è visualizzata nella figura 3.9. A titolo di esempio si mostra solo l'inserimento dì un data server di tecnologia Oracle.

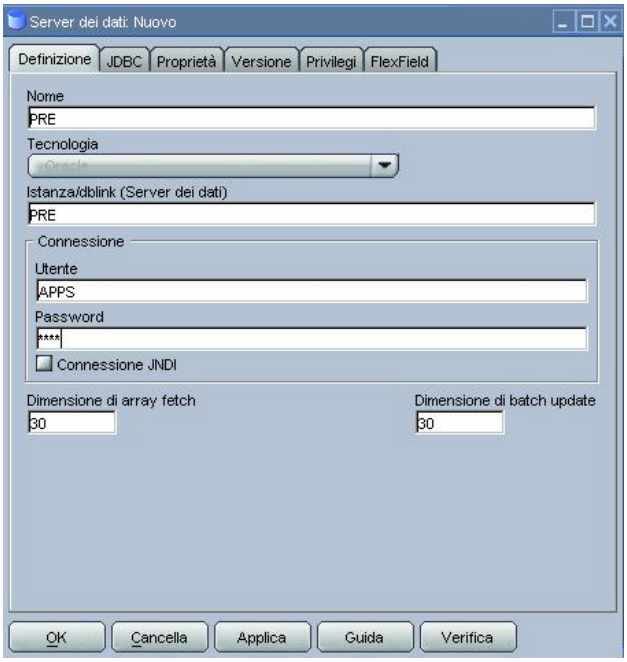


Figura 3.9: Creazione del data server Oracle

Sarà necessario riempire i seguenti campi:

- Nome: nome del data server
- Istanza/Dblink: nome fisico del server di dati, necessario in caso di creazione di un link fra due database
- Utente: utente abilitato alla lettura dei dati dalle tabelle del data server

- Password: password dell'utente
- Driver: (nella scheda accanto) driver e URL per la connessione al database

- **Creazione, per ciascun data server collegato, di uno o più schemi fisici associati**

  Nel momento in cui sarà creato il server dati, ODI mostrerà subito una nuova finestra con altri campi da compilare, relativi allo schema cui far riferimento per la connessione al database. I dettagli sono mostrati in figura 3.10.

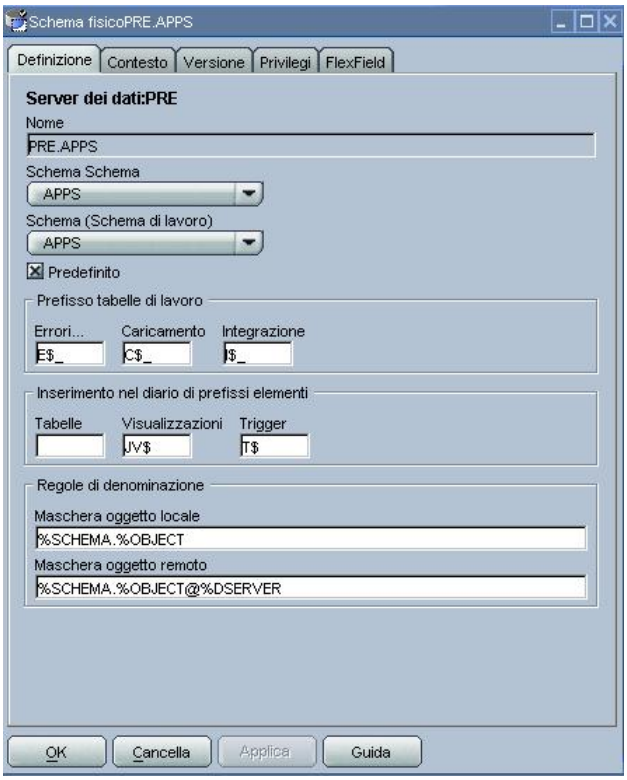


Figura 3.10: Creazione dello schema fisico per il data server

Sarà essenziale indicare:

- Schema: in cui Oracle Data Integrator cercherà le tabelle sorgenti o di destinazione

  - Schema di lavoro: in cui Oracle Data Integrator potrà creare o modificare strutture dati di lavoro temporanee associate alle tabelle sorgenti o di destinazione del data server
  - Contesto e schema logico: (nella scheda a fianco) contesto di esecuzione per il data server e schema logico associato a quello fisico di riferimento

- **Creazione degli schemi logici e associazione di ognuno di essi agli schemi fisici precedentemente indicati**

  A questo punto si dovrà procedere con la creazione di uno schema logico collegato allo schema fisico precedentemente definito. Per eseguire anche questo step di creazione della nostra topologia dovremo questa volta inserire solo un'informazione di cui ODI necessita in questa fase, ovvero l'indicazione di un contesto e di uno schema fisico da accoppiare a quello logico. (figura 3.11).

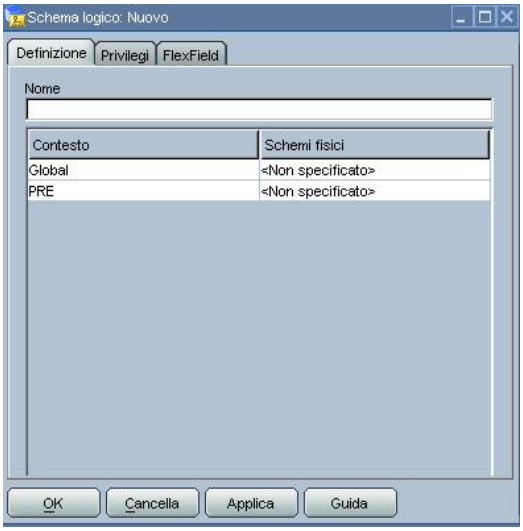


Figura 3.11: Creazione dello schema logico per il data server

- **Creazione di un agente fisico per ogni agente lanciato nella macchina (come listener)**

  L'agente è un servizio Java che può essere utilizzato come listener su una porta TCP/IP. Ciò permette di richiedere il lancio di interfacce, ses-

sioni scenari ecc., dai moduli grafici di ODI e di lanciare nei momenti prestabiliti uno scenario tramite la creazione di una schedulazione per l'agente stesso. Si mostra adesso come creare un nuovo agente fisico, con l'ausilio della figura 3.12.

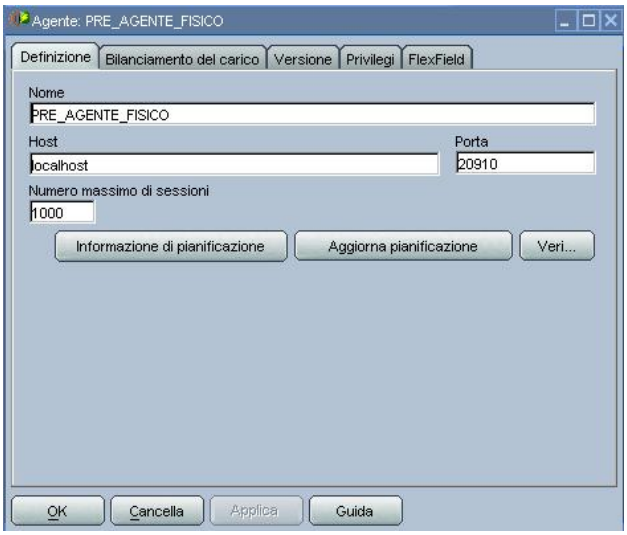


Figura 3.12: Creazione di un agente fisico

Le informazioni da riportare in questo caso sono:

- Nome: nome dell'agente
- Host: nome o indirizzo ip della macchina su cui verrà lanciato l'agente
- Porta: porta di ascolto utilizzata dall'agente, di default è la 20910
- numero massimo di sessioni: numero massimo di sessioni che l'agente potrà lanciare in contemporanea

- **Creazione degli agenti logici e loro associazione agli agenti fisici creati**

  Per ogni agente fisico ne dovrà essere indicato ovviamente uno logico. Questo è l'ultimo passo da effettuare per il completamento della definizione della topologia del nostro progetto. Vediamo quindi l'ultima finestra di inserimento (figura 3.13), il cui unico campo da indicare è l'agente

fisico che verrà accoppiato all'agente logico in un determinato contesto di esecuzione.

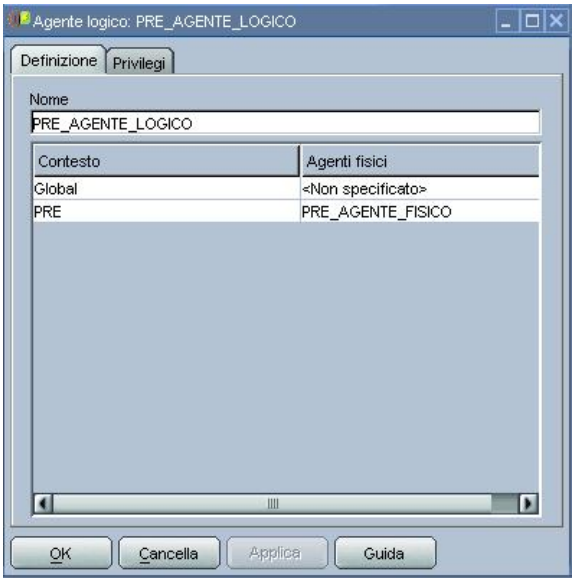


Figura 3.13: Creazione di un agente logico

3.3 L'implementazione del cubo Ore Lavorate

In questo paragrafo e nel successivo, ultimata la parte dedicata alla definizione della topologia del progetto, sarà trattato lo sviluppo delle due interfacce da realizzare per il caricamento dei dati nelle tabelle di destinazione. Il flusso ETL verrà implementato a partire dalle considerazioni già fatte nel capitolo della progettazione funzionale. Di conseguenza, in questa fase, si dovrà realizzare tecnicamente il passaggio dei dati dalle sorgenti alle destinazioni, seguendo il percorso stabilito in precedenza.

Prima di procedere, però, va fatta una precisazione doverosa. Nei paragrafi precedenti abbiamo accennato alla presenza di diversi knowledge module, che permettono di far interagire ODI con un gran numero di tecnologie, compreso Hyperion, il sistema in cui dovrebbero essere caricati i cubi da popolare. In realtà però, considerate le finalità soprattutto didattiche del presente progetto di tesi, le tabelle di destinazione saranno mantenute in un altro database Oracle. Questo comporta solo il trascuramento delle possibili migliorie a

livello prestazionale che dovrebbero essere ottenute migrando i dati analitici verso un sistema Hyperion. Quanto fatto consentirà comunque ugualmente di stabilire la validità dei risultati, in particolare perchè, parallelamente all'integrazione dei dati caricati nelle tabelle, saranno realizzati anche quei prospetti in Excel (e stavolta saranno utilizzati knowledge modules specifici per l'interazione con i file di tipo CSV), che possono essere letti in seguito da altri software per la creazione dei cubi.

Si inizia adesso la realizzazione del primo "package" di interfacce, dedicato al caricamento dei dati nel primo cubo delle ore lavorate. Come visto nella parte dedicata alla progettazione funzionale, la prima interfaccia da realizzare in questo caso è quella che va a popolare la tabella chiamata RICERCA_PROJECT, ovvero quella che ci consentirà di mantenere, per le successive fasi del flusso, soltanto quei record che rispettino i requisiti commissionati dal cliente, per le regole di business viste in precedenza. Per la costruzione di ogni interfaccia sarà necessario ogni volta eseguire determinati passi, dal caricamento delle tabelle source e target in ODI alla creazione del nuovo modello collegato ad uno schema fisico ecc. Essendo questa la prima interfaccia che viene analizzata, si andrà nel dettaglio a vedere come singolarmente queste operazioni vengano eseguite, mentre per le successive questi passaggi obbligatori verranno soltanto citati.

Per realizzare un interfaccia in ODI, ciò che inizialmente va fatto è concentrarsi sulle tabelle di input e di output che il suo flusso avrà come punto rispettivamente di inizio e di fine. Nel nostro caso le tabelle che vengono lette per recuperare i dati sono la tabella PA_PROJECTS_ALL e la vista ERS_PA_ORG_HIERARCHY_V (che si trovano all'interno del database PRE di Siram), mentre la tabella che verrà popolata è, come detto, RICERCA_PROJECT (tale tabella ancora non esiste, e va di conseguenza prima creata, con comandi SQL direttamente sul db che dovrà ospitarla, e definendo nome e datatype oppurtuni per ognuna delle colonne che la costituiscono). Innanzitutto queste tabelle vanno lette tramite i loro schemi, definiti precedentemente mediante il Topology Manager, (si è visto l'inserimento di quello dell'utente APPS), e solo a questo punto andrà creato in ODI un modello tramite cui sarà possibile eseguire il reverse engineering delle tabelle di cui si necessita. Vediamo, ad esempio, come far leggere ad ODI il contenuto della

tabella PA_PROJECTS_ALL, richiedendo l'inserimento di un nuovo modello per lo schema PA, proprietario della tabella in questione, come mostrato in figura 3.14.

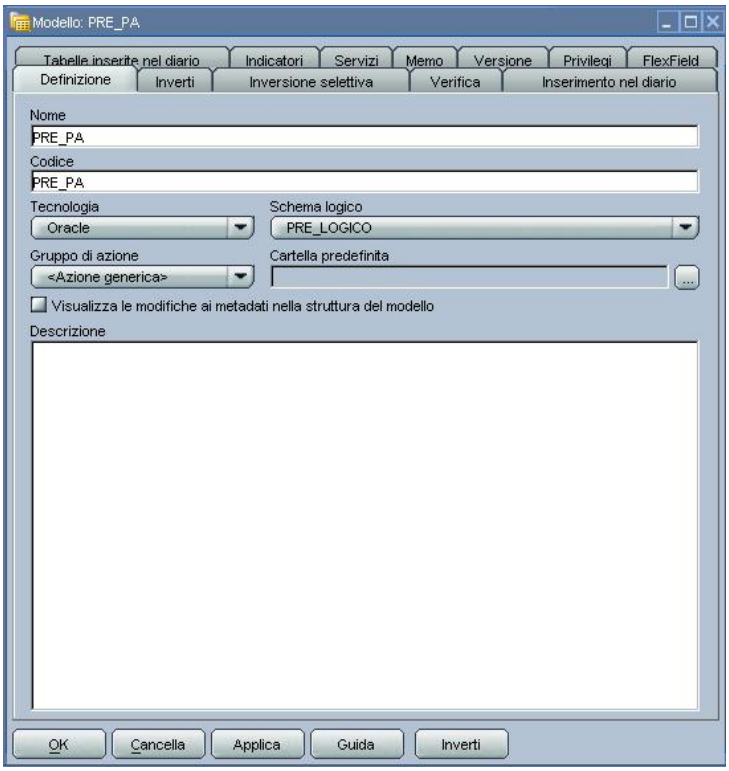


Figura 3.14: Creazione di un nuovo modello per lo schema PA

Inserendo nome, tecnologia e schema logico, sarà possibile aprire la scheda a fianco, chiamata "Inversione Selettiva". In questo caso, per visualizzare nel riquadro tutte le tabelle possedute dall'utente PA nel suo schema, si dovrà spuntare la casella "Oggetti da invertire". A questo punto si potranno selezionare a piacimento le tabelle da invertire, e, dando conferma, esse saranno caricate da ODI e mostrate all'interno del modello, come si vede nella schermata 3.15.

A questo punto sarà necessario ripetere la stessa procedura per la vista ERS_PA_ORG_HIERARCHY_V (schema APPS) e per la tabella di destinazione RICERCA_PROJECT (schema PROVA nel db che conterrà le tabelle target). Fatto ciò si potrà iniziare lo sviluppo dell'interfaccia vera e propria. Durante la sua creazione, affinchè l'esecuzione possa andare a buon fine, sarà

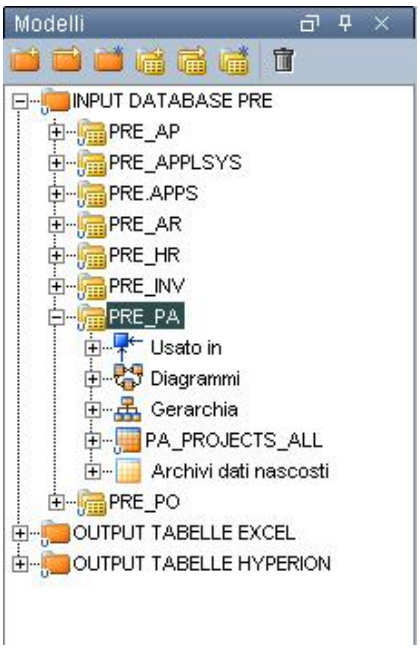


Figura 3.15: Tabella invertita all'interno del modello

necessario inserire tutte le informazioni riguardo flusso dei dati, collegamenti tra tabelle, restrinzioni e specifiche sugli knowledge module da utilizzare, nelle varie schede di cui la stessa si compone. Appena apparsa la form di creazione interfaccia, ODI chiede subito di impostare per essa nome, contesto di esecuzione e area di gestione temporanea, se diversa dal target, in cui ODI effettuerà quelle operazioni che non può eseguire direttamente quando legge i dati dalle tabelle sorgenti.

Una delle schede più importanti dell'interfaccia è sicuramente quella relativa al diagramma, in cui è necessario includere la tabelle sorgenti, esplicitando le relazione fra esse ed eventuali restrizioni, e la tabella di destinazione, per ogni colonna della quale va esplicitato il data mapping, ovvero va inserita la fonte da cui recuperare i dati. Le informazioni per i campi dell'archivio target potranno essere costruiti direttamente sulla sorgente, se essi vengono recuperati utilizzando una delle colonne delle tabelle source, o nell'area di gestione temporanea, negli altri casi. Ad esempio il mapping della colonna "Nome Commessa" sarà realizzato sulla sorgente, perchè il suo contenuto verrà riempito con i valori della colonna PA_PROJECTS_ALL.NAME solo per

quei record che vengono tenuti in seguito alle restrizioni, che sono state definite nel precedente capitolo sulla progettazione funzionale. Se invece, come per un campo della tabella di destinazione in cui va semplicemente memorizzata la data di estrazione, le informazioni non vengono recuperate da alcun campo delle tabelle sorgenti, allora bisognerà indicare ad ODI che la costruzione dello stesso dovrà avvenire nell'area di gestione temporanea, affinchè l'esecuzione dell'interfaccia vada a buon fine. Il diagramma relativo all'interfaccia sviluppata per popolare la tabella RICERCA_PROJECT è visualizzato in figura 3.16.

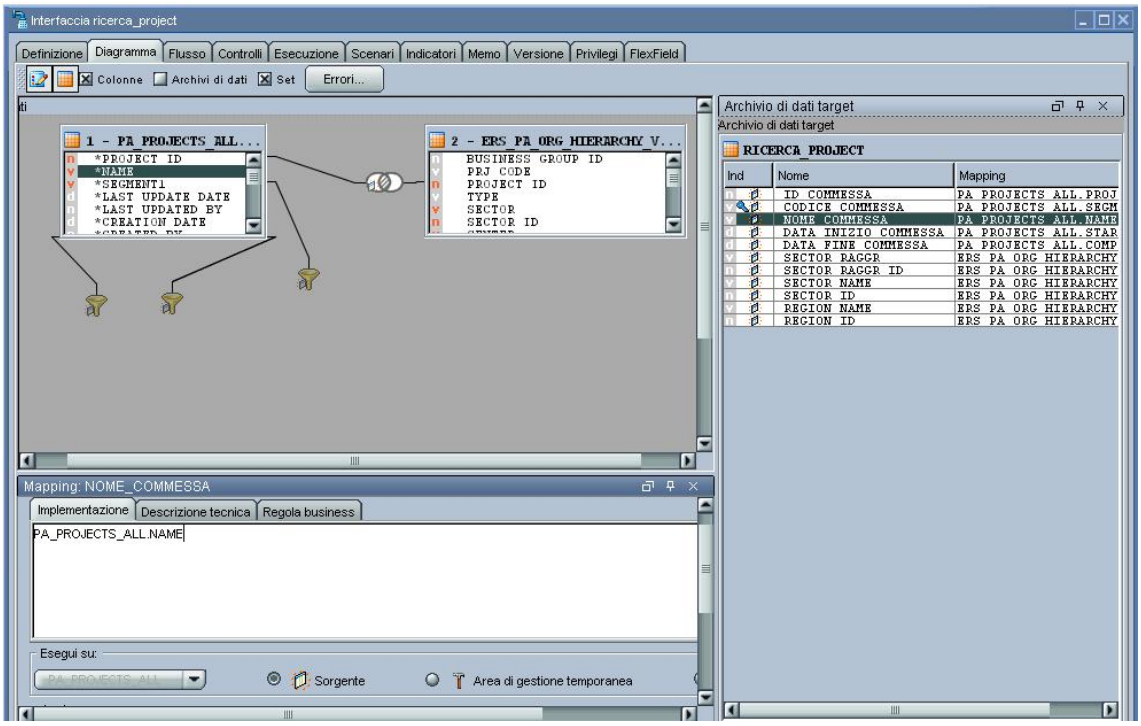


Figura 3.16: Creazione nuova interfaccia - Diagrammi

Le prossime due schede che compongono l'interfaccia servono in particolare a decidere i knowledge modules da utilizzare, in modo da abilitare la connessione alle tecnologie che sono coinvolte nel processo di ETL. La figura 3.17 permette l'inserimento delle informazioni relative ai knowledge module da scegliere per le fasi di caricamento e integrazione.

ODI in questa scheda riepiloga il flusso di dati risultante dalle informa-

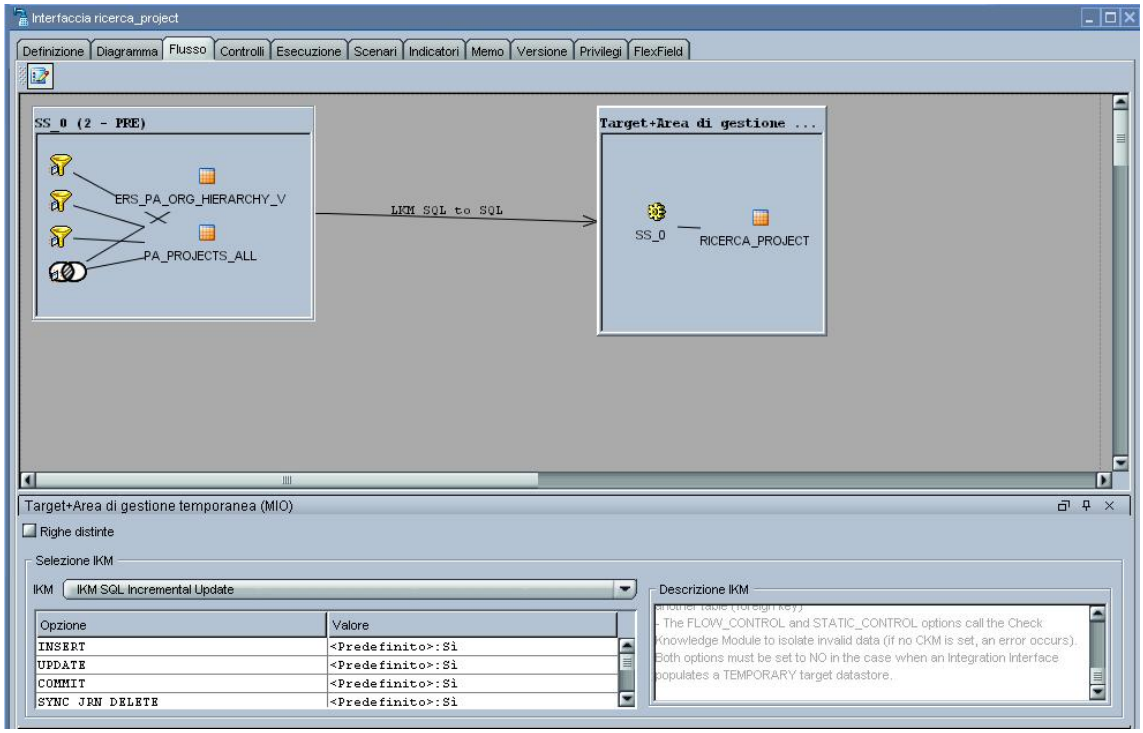


Figura 3.17: Creazione nuova interfaccia - Flusso

zioni inserite precedentemente, dividendolo in parte della sorgente e parte dell'area di gestione temporanea e del target. L'utente in questa sede dovrà, selezionando prima la form della sorgente e poi la form dell'area di gestione e del target, scegliere il knowledge module da utilizzare dal menu a discesa che viene visualizzato nella parte inferiore nel riquadro, e che permette inoltre di selezionare il valore dei parametri associati ai KM.

Nell'ultima scheda da considerare, ODI richiede di effettuare le stesse scelte relativamente però alla fase di controllo dei dati. I dettagli vengono mostrati in figura 3.18.

Anche in questo caso sarà presente, come si vede, un menu a discesa che consentirà la scelta del knowledge module desiderato. Completato anche questo step, sarà possibile eseguire l'interfaccia, ed osservare l'andamento del lancio tramite l'Operator. Per farlo ODI richiederà di indicare un contesto di esecuzione ed un agente fisico in precedenza definito, dopodichè restituirà un avviso che informa l'inizio dell'esecuzione del flusso di dati.

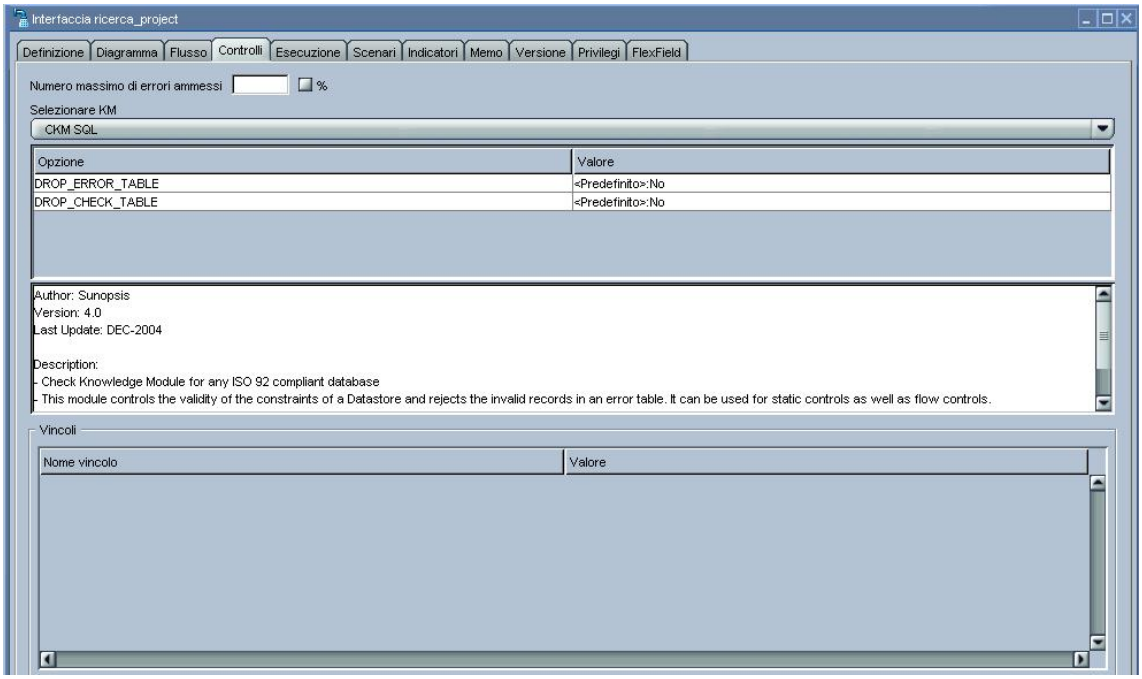


Figura 3.18: Creazione nuova interfaccia - Controlli

A questo punto, l'agente fisico lancerà il codice necessario per l'esecuzione dei vari passi che compongono le tre fasi di caricamento, controllo e integrazione dei dati. Come detto, per controllare l'esecuzione dell'interfaccia, sarà possibile utilizzare il modulo grafico Operator (figura 3.19)

Come si può vedere, questa volta ODI è riuscito a completare tutte le fasi dell'esecuzione, senza incontrare alcun tipo di problema. In ogni caso sarà comunque possibile controllare il codice SQL che è stato lanciato per effettuare qualunque passo fra quelli previsti. Solo a questo livello, quindi, l'utente si troverà di fronte quello in cui ODI ha trasformato le sue operazioni effettuate finora solo tramite i moduli grafici, a sottolineare il fatto che l'utente in questo caso, per lo sviluppo dei flussi ETL, può anche non disporre di competenze specifiche di programmazione.

Se non è stato ricevuto alcun messaggio di errore, la tabella target dovrebbe essere stata popolata con i record previsti. Potrà essere verificato il contenuto della tabella target selezionando la scelta dal menu relativo ad essa, all'interno del modello definito nel Designer, oppure cliccando col tasto

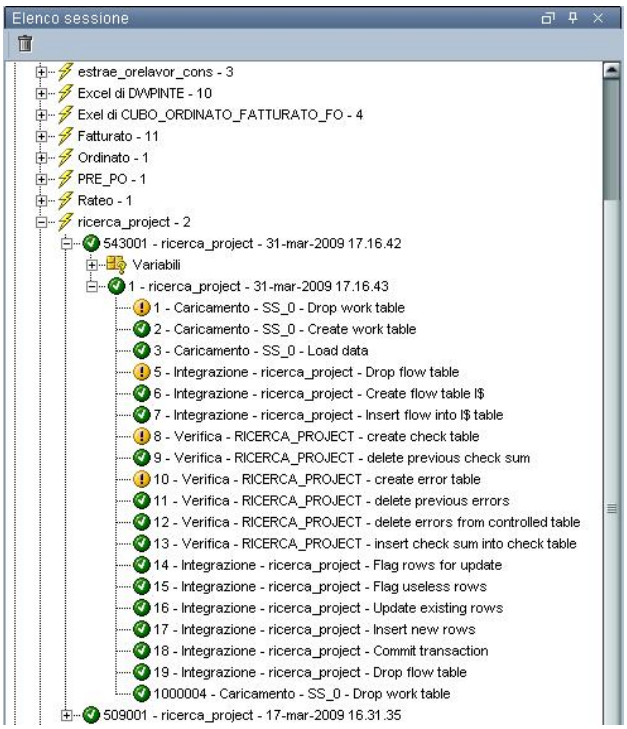


Figura 3.19: Operator - Esecuzione interfaccia RICERCA_PROJECT

destro nella scheda del diagramma dell'interfaccia relativa.

Popolata la tabella RICERCA_PROJECT , si potrà adesso procedere alla creazione di quelle che conterranno i dati aggregati, rispetto alle dimensioni considerate. La tabella appena realizzata farà parte delle sorgenti nel diagramma delle due interfacce che dovranno essere create. Iniziamo trattando il caricamento della tabella DWBUDPR, che contiene i dati delle ore lavorate previste dal budget iniziale per le commesse. Nel capitolo dedicato alla progettazione funzionale ritroviamo l'elenco delle tabelle sorgenti, del data mapping e delle business rules che riguardano la realizzazione dell'interfaccia relativa che deve essere creata in ODI. Alla fine, il risultato dell'inserimento dovrebbe portare ad avere un diagramma per l'interfaccia DWBUDPR simile a quello visualizzato in figura 3.20.

Le altre schede della form di creazione dell'interfaccia devono essere trattate in modo identico rispetto a quella realizzata in precedenza. Una volta inserite queste informazioni, essa potrà essere lanciata, e la sua esecuzio-

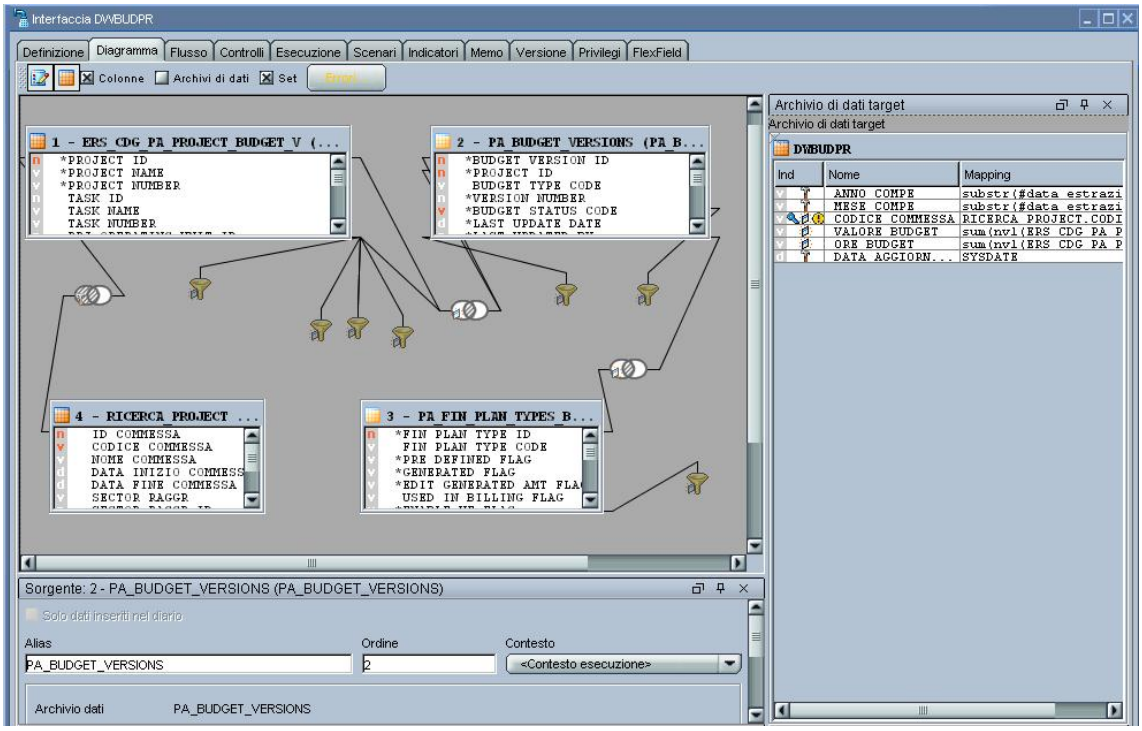


Figura 3.20: L'interfaccia per il caricamento di DWBUDPR

ne seguita tramite la solita finestra dell'Operator. Fatto ciò la tabella sarà popolata con i record previsti.

Per realizzare un'interfaccia che crei un' estrazione in Excel, invece che popolare la tabella di un database, sarà a questo punto sufficiente duplicare quella appena creata e modificare il knowledge module di integrazione, in maniera tale che i dati del flusso vengano indirizzati questa volta verso un file CSV, che verrà poi convertito nel formato desiderato. L'utente quindi, in questo caso, dovrà prima creare un file in formato CSV vuoto e permettere ad ODI di leggerlo e modificarlo, creando un apposito server di dati, come mostrato in figura 3.21.

Similmente a quanto visto nel paragrafo precedente, sarà necessario anche questa volta creare in aggiunta uno schema logico da accoppiare a quello fisico appena creato nel nostro contesto di esecuzione. Bisognerà inoltre creare dal Designer un modello adatto ad ospitare il file CSV, che ha caratteristiche diverse rispetto a quelle di una tabella in Oracle o Hyperion. Nella figura

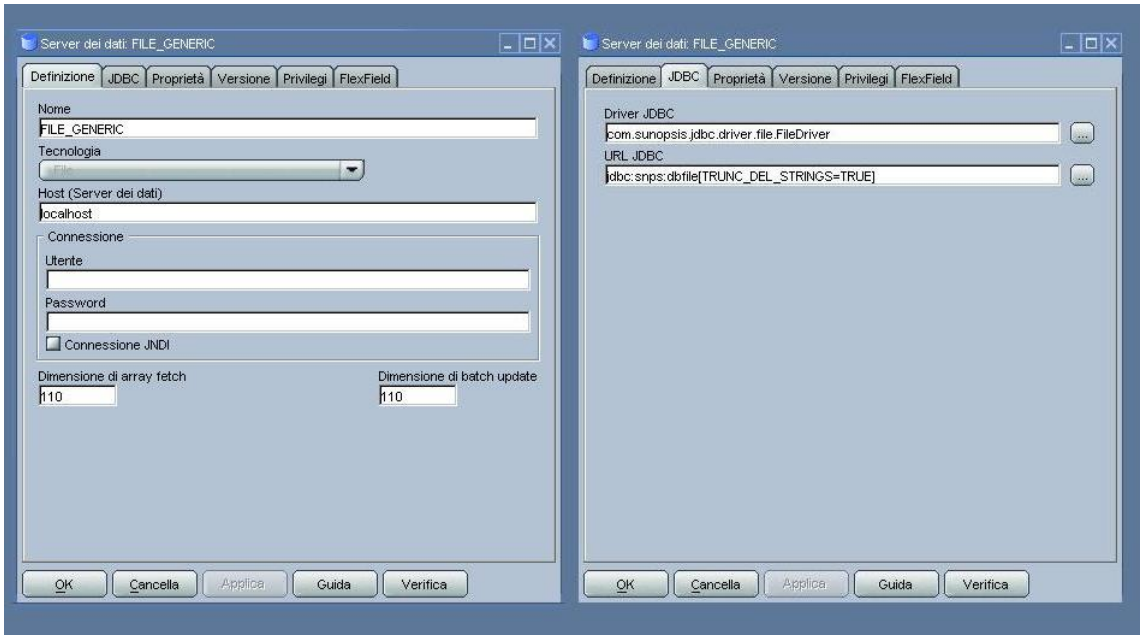


Figura 3.21: Creazione server e schema fisico del file

3.22 vengono visualizzati i dettagli da inserire per completare anche questo passo.

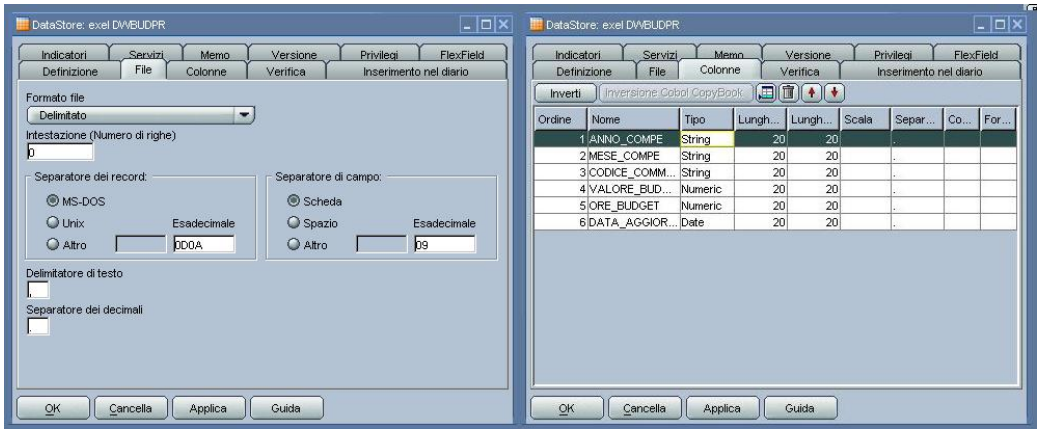
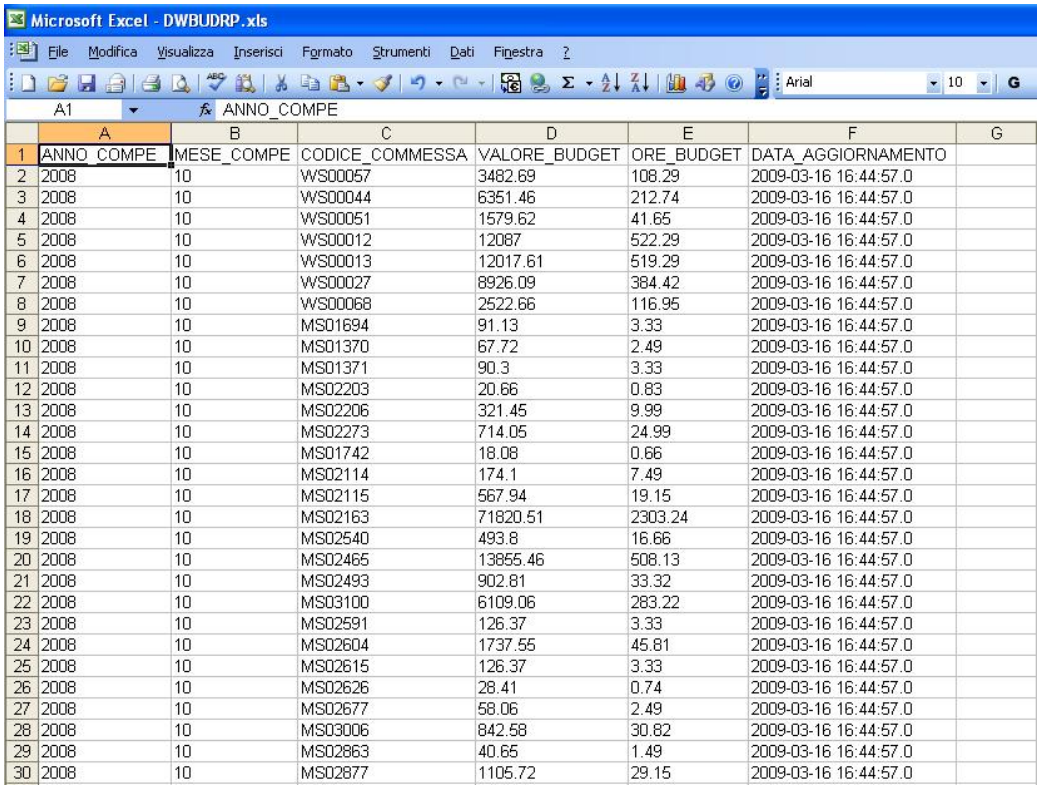


Figura 3.22: Creazione nuovo modello - file CSV

Nella prima scheda del modello sarà necessario in questo caso inserire il suo nome, tipo di archivio dati e il percorso per recuperare il file CSV vuoto precedentemente creato. Successivamente si dovrà impostare il file come di

tipo delimitato e inserire i caratteri che fungeranno per la delimitazione del testo e la separazione dei numeri decimali. Infine, nell'ultima parte considerata, dovranno essere inseriti i dettagli relativi alle colonne e i datatype che faranno parte della tabella.

Fatto ciò, l'interfaccia creata in precedenza dovrà essere semplicemente duplicata, cliccando col tasto destro sulla stessa e scegliendo l'opzione opportuna dal menu a discesa, provvedendo poi a modificare l'archivio target e il KM di integrazione (sarà scelto quello chiamato "SQL To File Append"). In tal modo il file CSV, creato vuoto in precedenza, verrà popolato, e, letto da Excel, restituirà i risultati mostrati in figura 3.23. Sembra scontato affermare che nel caso della programmazione tradizionale, per realizzare lo stesso scopo, i passi da effettuare sarebbero stati decisamente più lunghi e complessi.



Microsoft Excel - DWBUDRP.xls

File Modifica Visualizza Inserisci Formato Strumenti Dati Finestra ?

Arial 10 G

A1 ANNO_COMPE

	A	B	C	D	E	F	G
1	ANNO_COMPE	MESE_COMPE	CODICE_COMMESSA	VALORE_BUDGET	ORE_BUDGET	DATA_AGGIORNAMENTO	
2	2008	10	WS00057	3482.69	108.29	2009-03-16 16:44:57.0	
3	2008	10	WS00044	6351.46	212.74	2009-03-16 16:44:57.0	
4	2008	10	WS00051	1579.62	41.65	2009-03-16 16:44:57.0	
5	2008	10	WS00012	12087	522.29	2009-03-16 16:44:57.0	
6	2008	10	WS00013	12017.61	519.29	2009-03-16 16:44:57.0	
7	2008	10	WS00027	8926.09	384.42	2009-03-16 16:44:57.0	
8	2008	10	WS00068	2522.66	116.95	2009-03-16 16:44:57.0	
9	2008	10	MS01694	91.13	3.33	2009-03-16 16:44:57.0	
10	2008	10	MS01370	67.72	2.49	2009-03-16 16:44:57.0	
11	2008	10	MS01371	90.3	3.33	2009-03-16 16:44:57.0	
12	2008	10	MS02203	20.66	0.83	2009-03-16 16:44:57.0	
13	2008	10	MS02206	321.45	9.99	2009-03-16 16:44:57.0	
14	2008	10	MS02273	714.05	24.99	2009-03-16 16:44:57.0	
15	2008	10	MS01742	18.08	0.66	2009-03-16 16:44:57.0	
16	2008	10	MS02114	174.1	7.49	2009-03-16 16:44:57.0	
17	2008	10	MS02115	567.94	19.15	2009-03-16 16:44:57.0	
18	2008	10	MS02163	71820.51	2303.24	2009-03-16 16:44:57.0	
19	2008	10	MS02540	493.8	16.66	2009-03-16 16:44:57.0	
20	2008	10	MS02465	13855.46	508.13	2009-03-16 16:44:57.0	
21	2008	10	MS02493	902.81	33.32	2009-03-16 16:44:57.0	
22	2008	10	MS03100	6109.06	283.22	2009-03-16 16:44:57.0	
23	2008	10	MS02591	126.37	3.33	2009-03-16 16:44:57.0	
24	2008	10	MS02604	1737.55	45.81	2009-03-16 16:44:57.0	
25	2008	10	MS02615	126.37	3.33	2009-03-16 16:44:57.0	
26	2008	10	MS02626	28.41	0.74	2009-03-16 16:44:57.0	
27	2008	10	MS02677	58.06	2.49	2009-03-16 16:44:57.0	
28	2008	10	MS03006	842.58	30.82	2009-03-16 16:44:57.0	
29	2008	10	MS02863	40.65	1.49	2009-03-16 16:44:57.0	
30	2008	10	MS02877	1105.72	29.15	2009-03-16 16:44:57.0	

Figura 3.23: Porzione dei record del file excel DWBUDPR

Per la realizzazione del cubo finale, è necessario però anche, come visto nel documento funzionale, il caricamento di un'altra tabella, chiamata DWPIN-

TE che ricapitolerà, sotto diversi aspetti, le ore consuntive, ovvero quelle che effettivamente si sono verificate nel periodo di estrazione considerato. Anche in questo caso, nella scheda del diagramma dell'interfaccia relativa da creare, viene utilizzata fra le tabelle sorgenti la RICERCA_PROJECT, che mantiene solo i record da tenere in considerazione, come si vede in figura 3.24.

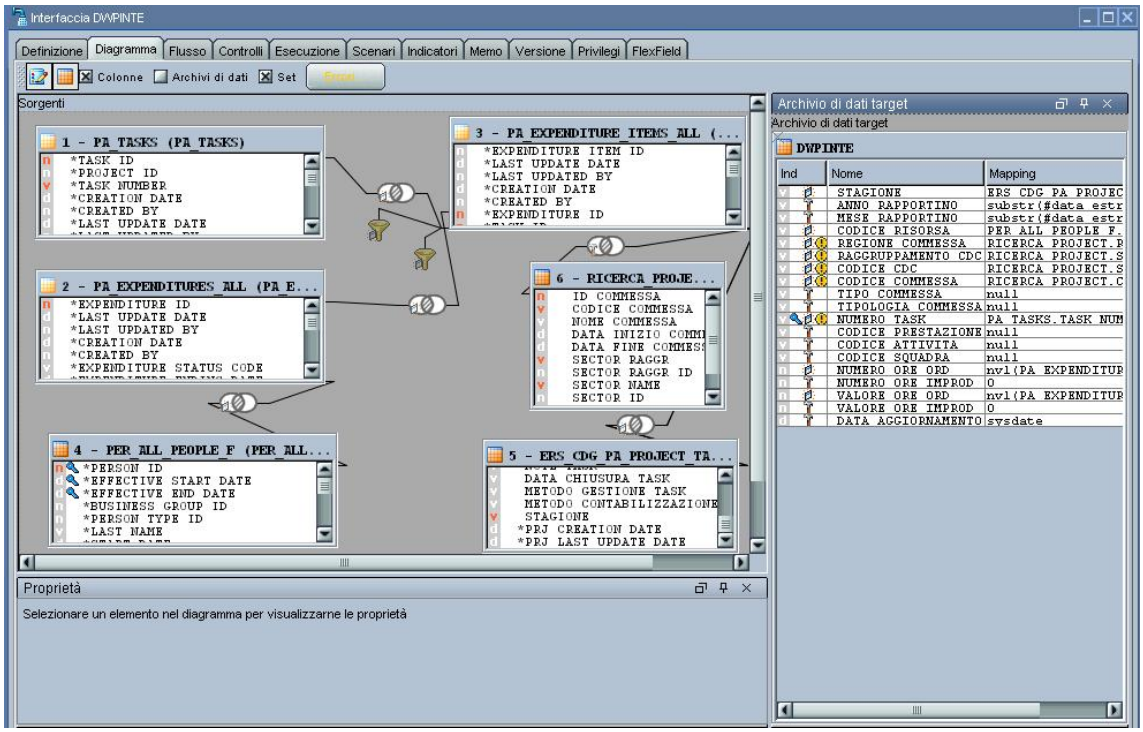
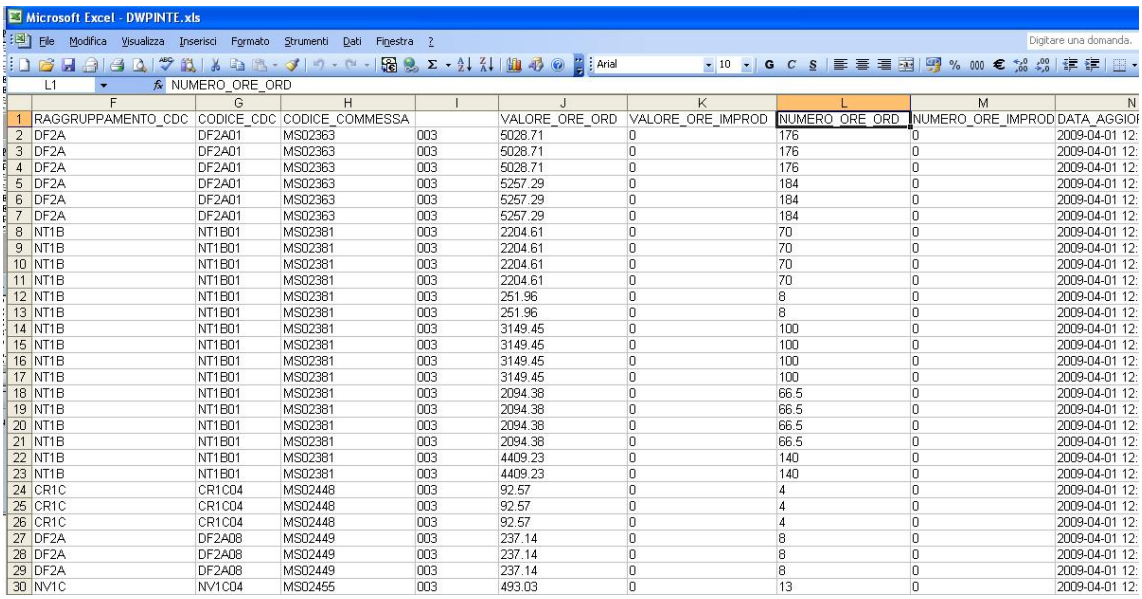


Figura 3.24: Interfaccia della tabella DWPINTE

Si ribadisce l'importanza di curare con la massima attenzione questa parte dell'interfaccia, poichè rappresenta quella in cui effettivamente, definendo filtri e collegamenti (join) tra le tabelle, oltre a tabelle sorgenti e target, verrà definito il flusso di dati che andrà poi a caricare la tabella finale. L'interfaccia dovrà essere, allo stesso modo di prima, duplicata in maniera tale da produrre anche il prospetto Excel che ospiterà gli stessi record, modificando semplicemente il knowledge module riferito alla fase di integrazione. In figura 3.25 viene mostrata una porzione di record presenti all'interno del file creato.



Microsoft Excel - DWPINTE.xls

File Modifica Visualizza Inserisci Formato Strumenti Dati Finestra ?

L1 NUMERO_ORE_ORD

	F	G	H	I	J	K	L	M	N
1	RAGGRUPPAMENTO_CDC	CODICE_CDC	CODICE_COMMESSA		VALORE_ORE_ORD	VALORE_ORE_IMPROD	NUMERO_ORE_ORD	NUMERO_ORE_IMPROD	DATA_AGGIOI
2	DF2A	DF2A01	MS02363	003	5028.71	0	176	0	2009-04-01 12:
3	DF2A	DF2A01	MS02363	003	5028.71	0	176	0	2009-04-01 12:
4	DF2A	DF2A01	MS02363	003	5028.71	0	176	0	2009-04-01 12:
5	DF2A	DF2A01	MS02363	003	5257.29	0	184	0	2009-04-01 12:
6	DF2A	DF2A01	MS02363	003	5257.29	0	184	0	2009-04-01 12:
7	DF2A	DF2A01	MS02363	003	5257.29	0	184	0	2009-04-01 12:
8	NT1B	NT1B01	MS02381	003	2204.61	0	70	0	2009-04-01 12:
9	NT1B	NT1B01	MS02381	003	2204.61	0	70	0	2009-04-01 12:
10	NT1B	NT1B01	MS02381	003	2204.61	0	70	0	2009-04-01 12:
11	NT1B	NT1B01	MS02381	003	2204.61	0	70	0	2009-04-01 12:
12	NT1B	NT1B01	MS02381	003	251.96	0	8	0	2009-04-01 12:
13	NT1B	NT1B01	MS02381	003	251.96	0	8	0	2009-04-01 12:
14	NT1B	NT1B01	MS02381	003	3149.45	0	100	0	2009-04-01 12:
15	NT1B	NT1B01	MS02381	003	3149.45	0	100	0	2009-04-01 12:
16	NT1B	NT1B01	MS02381	003	3149.45	0	100	0	2009-04-01 12:
17	NT1B	NT1B01	MS02381	003	3149.45	0	100	0	2009-04-01 12:
18	NT1B	NT1B01	MS02381	003	2094.38	0	66.5	0	2009-04-01 12:
19	NT1B	NT1B01	MS02381	003	2094.38	0	66.5	0	2009-04-01 12:
20	NT1B	NT1B01	MS02381	003	2094.38	0	66.5	0	2009-04-01 12:
21	NT1B	NT1B01	MS02381	003	2094.38	0	66.5	0	2009-04-01 12:
22	NT1B	NT1B01	MS02381	003	4409.23	0	140	0	2009-04-01 12:
23	NT1B	NT1B01	MS02381	003	4409.23	0	140	0	2009-04-01 12:
24	CR1C	CR1C04	MS02448	003	92.57	0	4	0	2009-04-01 12:
25	CR1C	CR1C04	MS02448	003	92.57	0	4	0	2009-04-01 12:
26	CR1C	CR1C04	MS02448	003	92.57	0	4	0	2009-04-01 12:
27	DF2A	DF2A08	MS02449	003	237.14	0	8	0	2009-04-01 12:
28	DF2A	DF2A08	MS02449	003	237.14	0	8	0	2009-04-01 12:
29	DF2A	DF2A08	MS02449	003	237.14	0	8	0	2009-04-01 12:
30	NV1C	NV1C04	MS02455	003	493.03	0	13	0	2009-04-01 12:

Figura 3.25: Porzione dei record del file excel DWPINTE

3.4 L'implementazione del cubo Ordini Materiali

In questo paragrafo verrà trattato il caricamento della tabella e del file Excel che serviranno per la realizzazione del cubo Ordini Materiali. Come descritto nella parte funzionale, anche in questo caso, dovranno essere popolate alcune tabelle intermedie per raggiungere il risultato finale, e ciò comporterà dunque la realizzazione di un diverso numero di interfacce, per le quali andranno sostanzialmente ripetuti tutti i passi visti in dettaglio nel paragrafo precedente, che saranno trattati in questa parte più velocemente, per non dilungare oltremodo la trattazione.

La prima cosa che viene fatta, in maniera similare a quanto visto nel precedente capitolo, è memorizzare in un tabella temporanea solo quei record relativi agli ordini che soddisfano le business rules richieste dal cliente. Ovviamente sarà necessaria la creazione di una nuova interfaccia, in cui impostare filtri e data mapping in maniera opportuna, cosicchè, a partire dalle tabelle sorgenti indicate, vengano recuperati solo i record significativi da includere nelle indagini da analizzare.

Soltanto i record mantenuti in questa tabella verranno utilizzati per calcolare i tre campi aggregati del rapporto finale. Come visto nel documento

funzionale, il successivo step di integrazione prevede in questo caso la creazione di altre tre tabelle intermedie che consentiranno di crare ognuna uno dei valori numerici. Tali tabelle, chiamate Ordinato, Fatturato e Rateo, vengono create in maniera quasi identica, raggruppando i dati per le quattro dimensioni considerate e sommando il campo di una delle tabelle sorgenti che contiene il valore ricercato, implementando però, per ognuna, ogni volta delle business rules diverse (ciò richiede la creazione di tre interfacce differenti). Per motivi di brevità andremo soltanto a visualizzare l'interfaccia di Ordinato, poichè per le restanti due l'implementazione è molto simile. Lo schema viene mostrato in figura 3.26.

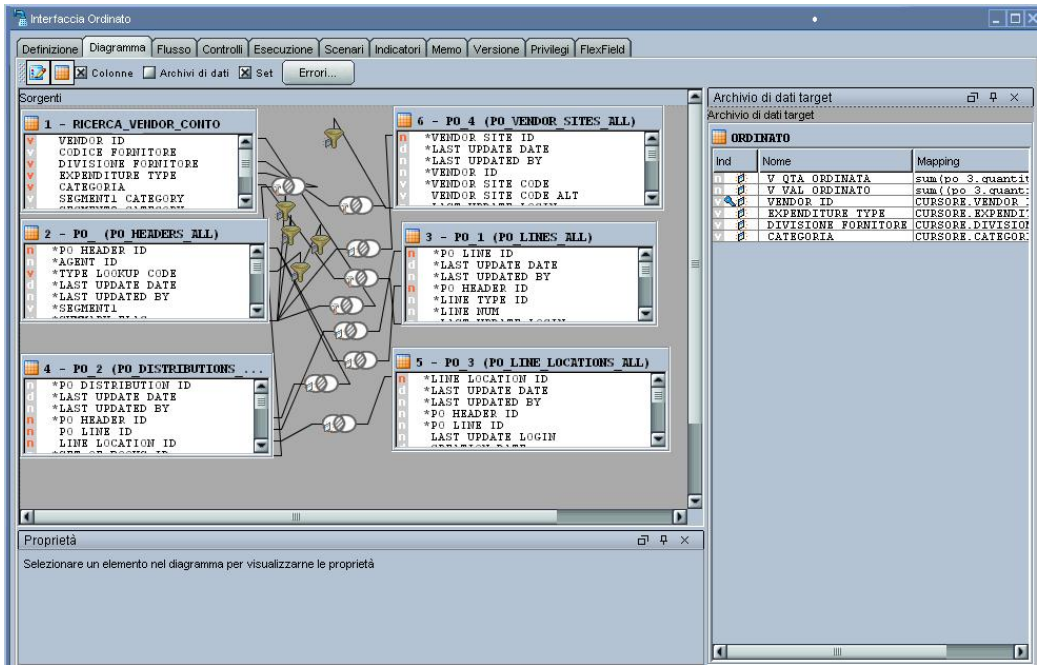


Figura 3.26: Interfaccia della tabella Ordinato

Una volta create queste tre tabelle, l'ultimo step del flusso ETL per la realizzazione della tabella finale DWFATVE2, e del corrispondente file Excel, che serviranno per la creazione del cubo delle richieste di approvvigionamento, tramite cui sarà possibile effettuare le indagini sugli importi degli ordini dal punto di vista di qualsiasi delle dimensioni considerate, prevede il collegamento fra gli archivi finora creati e in aggiunta la tabella che permette di recuperare il conto dell'ordine considerato. Fatto ciò si avrà, infatti, che

per ogni record presente nella tabella finale, rappresentante uno degli ordini di cui il cliente vuole calcolare le informazioni aggregate, verrà visualizzato importo ordinato, fatturato e valore del rateo raggruppando per tutte le dimensioni considerate in precedenza. Disponendo di queste informazioni, sarà possibile in seguito, sfruttando gli operatori OLAP, navigare il cubo raggruppando i dati per ognuno dei campi discreti singolarmente, in modo da avere il dettaglio degli importi versati da tutti i punti di vista possibili.

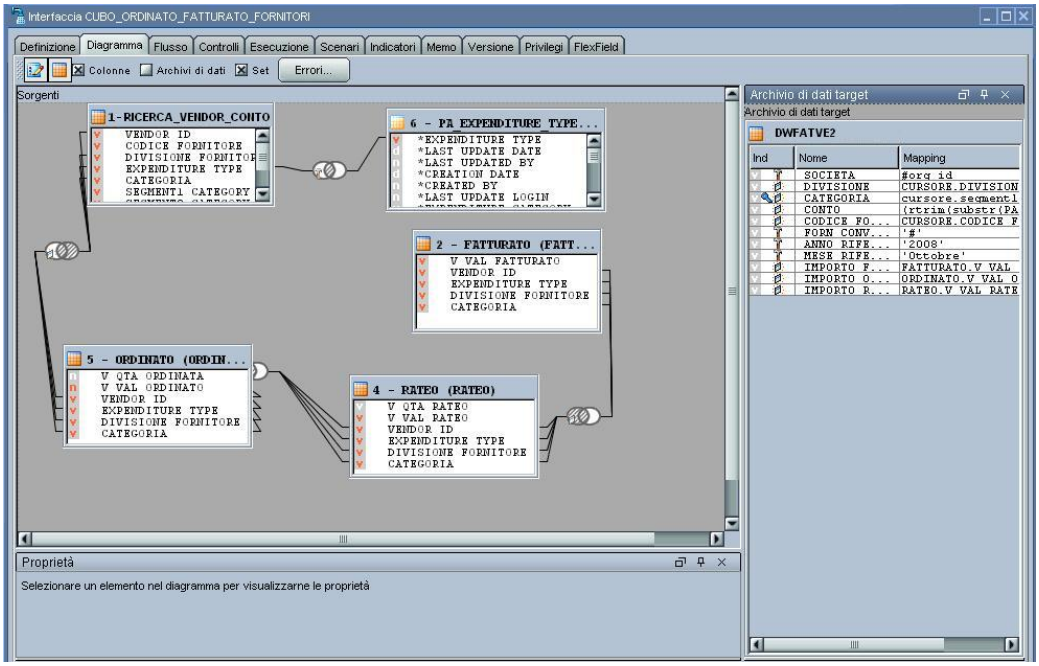
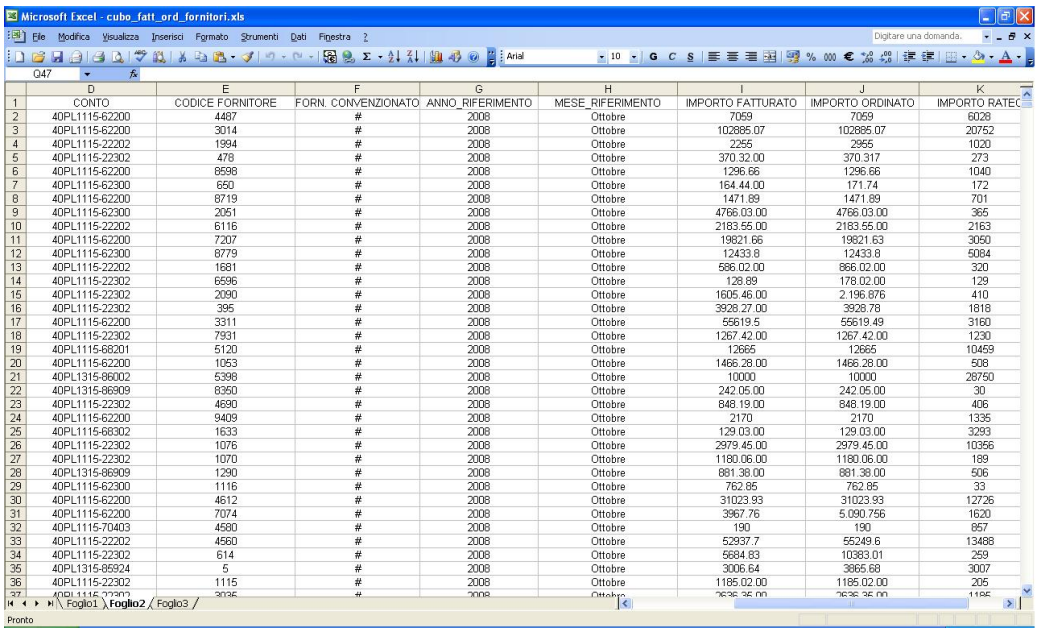


Figura 3.27: Interfaccia della tabella DWFATVE2

Anche per quest'ultimo caso viene mostrato il diagramma relativo all'interfaccia che viene creata, tramite la figura 3.27.

Al solito, essendo richiesto che gli stessi record compaiano anche nel file Excel, è necessario la duplicazione dell'interfaccia e la modifica del KM dedicato alla fase di integrazione. Successivamente, fatta partirne l'esecuzione, essa popolerà il file CSV, in precedenza creato vuoto, che, una volta importato correttamente in Excel, darà luogo al risultato mostrato in figura 3.28.



	D	E	F	G	H	I	J	K
1	CONTO	CODICE FORNITORE	FORN. CONVENZIONATO	ANNO_RIFERIMENTO	MESE_RIFERIMENTO	IMPORTO FATTURATO	IMPORTO ORDINATO	IMPORTO RATEC
2	40PL1115-62200	4487	#	2008	Ottobre	7059	7059	6028
3	40PL1115-62200	3014	#	2008	Ottobre	102885.07	102885.07	20752
4	40PL1115-22202	1994	#	2008	Ottobre	2255	2955	1020
5	40PL1115-22302	478	#	2008	Ottobre	370.32.00	370.317	273
6	40PL1115-62200	8598	#	2008	Ottobre	1296.66	1296.66	1040
7	40PL1115-62300	650	#	2008	Ottobre	164.44.00	171.74	172
8	40PL1115-62200	8719	#	2008	Ottobre	1471.89	1471.89	701
9	40PL1115-62300	2051	#	2008	Ottobre	4766.03.00	4766.03.00	365
10	40PL1115-22202	6116	#	2008	Ottobre	2183.55.00	2183.55.00	2163
11	40PL1115-62200	7207	#	2008	Ottobre	19821.66	19821.63	3050
12	40PL1115-62300	8779	#	2008	Ottobre	12433.8	12433.8	5084
13	40PL1115-22202	1681	#	2008	Ottobre	586.02.00	866.02.00	320
14	40PL1115-22302	6596	#	2008	Ottobre	128.89	178.02.00	129
15	40PL1115-22302	2090	#	2008	Ottobre	1605.46.00	2.196.876	410
16	40PL1115-22302	395	#	2008	Ottobre	3928.27.00	3928.78	1818
17	40PL1115-62200	3311	#	2008	Ottobre	55619.5	55619.49	3160
18	40PL1115-22302	7931	#	2008	Ottobre	1267.42.00	1267.42.00	1230
19	40PL1115-68201	5120	#	2008	Ottobre	12665	12665	10459
20	40PL1115-62200	1053	#	2008	Ottobre	1466.28.00	1466.28.00	508
21	40PL1315-86002	5398	#	2008	Ottobre	10000	10000	28750
22	40PL1315-86909	8350	#	2008	Ottobre	242.05.00	242.05.00	30
23	40PL1115-22302	4690	#	2008	Ottobre	848.19.00	848.19.00	406
24	40PL1115-62200	9409	#	2008	Ottobre	2170	2170	1335
25	40PL1115-68302	1633	#	2008	Ottobre	129.03.00	129.03.00	3293
26	40PL1115-22302	1076	#	2008	Ottobre	2979.45.00	2979.45.00	10356
27	40PL1115-22302	1070	#	2008	Ottobre	1180.06.00	1180.06.00	189
28	40PL1315-86909	1290	#	2008	Ottobre	881.38.00	881.38.00	506
29	40PL1115-62300	1116	#	2008	Ottobre	762.85	762.85	33
30	40PL1115-62200	4612	#	2008	Ottobre	31023.93	31023.93	12726
31	40PL1115-62200	7074	#	2008	Ottobre	3967.76	5.090.756	1620
32	40PL1115-70403	4580	#	2008	Ottobre	190	190	857
33	40PL1115-22202	4560	#	2008	Ottobre	52937.7	55249.6	13488
34	40PL1115-22302	614	#	2008	Ottobre	5684.83	10383.01	259
35	40PL1315-85924	5	#	2008	Ottobre	3006.64	3865.68	3007
36	40PL1115-22302	1115	#	2008	Ottobre	1185.02.00	1185.02.00	205

Figura 3.28: Porzione dei record del file excel DWFATVE2

3.5 Implementazione delle modifiche ai requisiti

Nel capitolo sulla progettazione funzionale è stato dedicato un paragrafo alla trattazione del caso di modifica dei requisiti commissionati inizialmente dal cliente per la realizzazione del cubo, la qual cosa andava ovviamente ad incidere anche sulla fase di integrazione dei dati, trattata nel paragrafo immediatamente precedente. Le modifiche consistevano nell'aggiunta di una dimensione sia per il cubo delle ore lavorate (mansione dell'impiegato), sia per quello degli acquisti dai fornitori (città del venditore). Lo scopo era quello di valutare la facilità di impatto delle modifiche sia nel caso dell'integrazione con ODI che nel caso dei programmi scritti in PL/SQL, in maniera tale da aggiungere altri termini di paragone al confronto critico tra queste due soluzioni.

Ciò che ci si propone quindi in questo paragrafo è di valutare le modifiche alle interfacce che devono essere effettuate per raggiungere il risultato desiderato. Per ciò che riguarda il caso dell'integrazione della tabella che servirà per la realizzazione del cubo acquisti, come detto anche in precedenza, l'impatto consiste nell'aggiunta di un campo in due interfacce che fanno parte del-

l'intero processo di ETL, quella della tabella RICERCA_VENDOR_CONTO e quella della tabella DWFATVE2. In entrambi i casi ciò che andrà fatto sarà innanzitutto, tramite comandi sql e nel db che le contiene, inserire una nuova colonna per entrambe le tabelle, che serviranno ad ospitare il nuovo campo Citta_fornitore. In seguito si dovranno modificare le vecchie interfacce, inserendo nel diagramma il nuovo archivio target e rieseguendo il data mapping, includendo l'implementazione anche del nuovo campo da considerare. Infine esse andranno eseguite nuovamente; questo processo andrà ovviamente effettuato prima per l'interfaccia di RICERCA_VENDOR_CONTO e poi per quella di DWFATVE2.

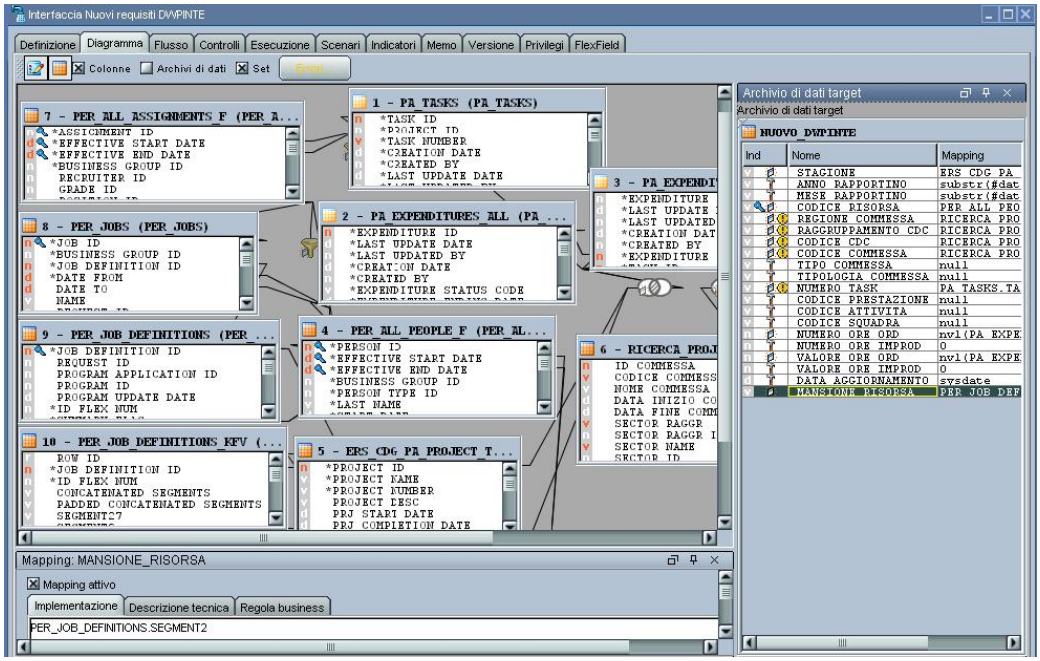


Figura 3.29: Interfaccia della tabella DWFATVE2, dopo cambio requisiti

Le modifiche da effettuare, nel caso del cubo delle ore lavorate, invece, risultano decisamente più consistenti. Per recuperare il campo che descrive la mansione di ogni employee da considerare (quello che ha lavorato, generando il costo nel periodo di estrazione), è necessario infatti considerare altre tabelle che permettono di recuperare le informazioni richieste e consentono inoltre di effettuare i controlli per fare in modo che per ogni employee venga estratto un solo record, come imposto anche tramite le business rules viste nel capitolo

della progettazione funzionale. Il risultato è mostrato in figura 3.29, in cui si mostra anche l'implementazione del nuovo campo.

Nel caso della realizzazione del prospetto Excel, che ospiti anch'esso le nuove modifiche ed il nuovo campo relativo alla mansione della risorsa, non si dovrà ovviamente aggiungere la nuova colonna utilizzando comandi SQL, poichè il file da popolare è di tipo CSV. In questo caso le intestazioni dei campi vengono aggiunte direttamente nella fase di integrazione, utilizzando il giusto knowledge module, rispettando le indicazioni contenute all'interno del modello dedicato al caricamento del file. Quindi basterà limitarsi ad aggiungere una nuova colonna utilizzando il puntatore e le modifiche aggiorneranno automaticamente l'archivio dati target dell'interfaccia relativa, nel momento in cui quest'ultima sarà riaperta per la prima volta dopo le modifiche effettuate.

Di seguito, in figura 3.30, si mostrano alcuni dei record contenuti nel file excel in tal modo creato.

	STAGIONE	ANNO_RAPP...	MESE_RAP...	CODICE_RISORSA	MANSIONE_RISORSA	NUMERO_ORE_ORD	VALORE_ORE_ORD
61	1994	2008	10	I001007	IMPIEGATO TECNICO/DIRETTIVO	0,5	15,21
62	1994	2008	10	I001007	IMPIEGATO TECNICO/DIRETTIVO	0,75	22,82
63	1994	2008	10	I001007	IMPIEGATO TECNICO/DIRETTIVO	2,25	68,45
64	1994	2008	10	I001007	IMPIEGATO TECNICO/DIRETTIVO	107	3.254,98
65	1994	2008	10	I001010	SEGRETERIA TECN/AMM.VA	25,75	783,32
66	1994	2008	10	I001615	CONDUTTORE/MANUTENTORE IMP. FRI	0	0
67	1994	2008	10	I001616	FUOCHISTA 1 GRADO	181	4.188,75
68	1994	2008	10	I001627	GENERICO	176	5.028,71
69	1994	2008	10	I001628	MANUTENTORE FRIGOTERMICO	4	109,36
70	1994	2008	10	I001628	MANUTENTORE FRIGOTERMICO	139	3.800,13
71	1994	2008	10	I000066	COORDINATORE CONTRATTI	184	7.530,91
72	1994	2008	10	I000069	CONTRACT ADMINISTRATOR	176	4.972,97
73	1994	2008	10	I000146	COORDINATORE CONTRATTI	168	4.746,93
74	1994	2008	10	I000153	NON DEFINITO	184	6.978,23
75	1994	2008	10	I000155	NON DEFINITO	136	5.157,82
76	1994	2008	10	I000200	NON DEFINITO	168	5.110,62
77	1994	2008	10	I000201	SEGRETERIA TECN/AMM.VA	168	5.110,62
78	1994	2008	10	I000216	COORDINATORE CONTRATTI	0	0
79	1994	2008	10	I000249	CAPO SQUADRA	156	5.017,04
80	1994	2008	10	I000335	SEGRETERIA TECN/AMM.VA	192	5.987,01
81	1994	2008	10	I000337	SEGRETERIA TECN/AMM.VA	207	6.454,74
82	1994	2008	10	I000361	CAPO SQUADRA	168	5.402,96
83	1994	2008	10	I000387	NON DEFINITO	152	5.764,62
84	1994	2008	10	I000388	TECNICO PREVENTIVISTA	175	4.741,55
85	1994	2008	10	I000390	CAPO SQUADRA	130	3.544,78
86	1994	2008	10	I000390	CAPO SQUADRA	30	818,03
87	1994	2008	10	I000399	SEGRETERIA TECN/AMM.VA	72	2.315,56

Figura 3.30: Porzione di record e campi della tabella NUOVO_DWFATVE2

3.6 Il rilascio in produzione: package e scenari

Una volta che sono state implementate tutte le interfacce necessarie, l'ultimo passo da effettuare per rendere operativo e pronto per l'avvio in produzione

l'intero processo che effettua il caricamento delle tabelle necessarie alla creazione dei cubi, consiste nella loro inclusione in package e nella creazione degli scenari relativi, uno per ogni cubo che verrà poi realizzato. In questo paragrafo verranno illustrati i passaggi relativi all'esecuzione di quest'ultimo step di integrazione, soltanto per ciò che concerne il cubo degli acquisti dai fornitori, per motivi di brevità e anche perchè i passaggi da compiere nel caso del cubo delle ore lavorate sarebbero comunque molto simili.

Lo scopo della creazione del package è di definire l'intero workflow per il caricamento della tabella DWFATVE2 e di stabilire l'esatta sequenza di esecuzione delle interfacce che permettono il suo caricamento. Questo viene fatto tramite la creazione di un nuovo package, mediante il Designer. Dopo avergli dato un nome, si dovrà definire il diagramma di cui Oracle Data Integrator dovrà tener conto per la sequenza del lancio di interfacce, come viene visualizzato in figura 3.31.

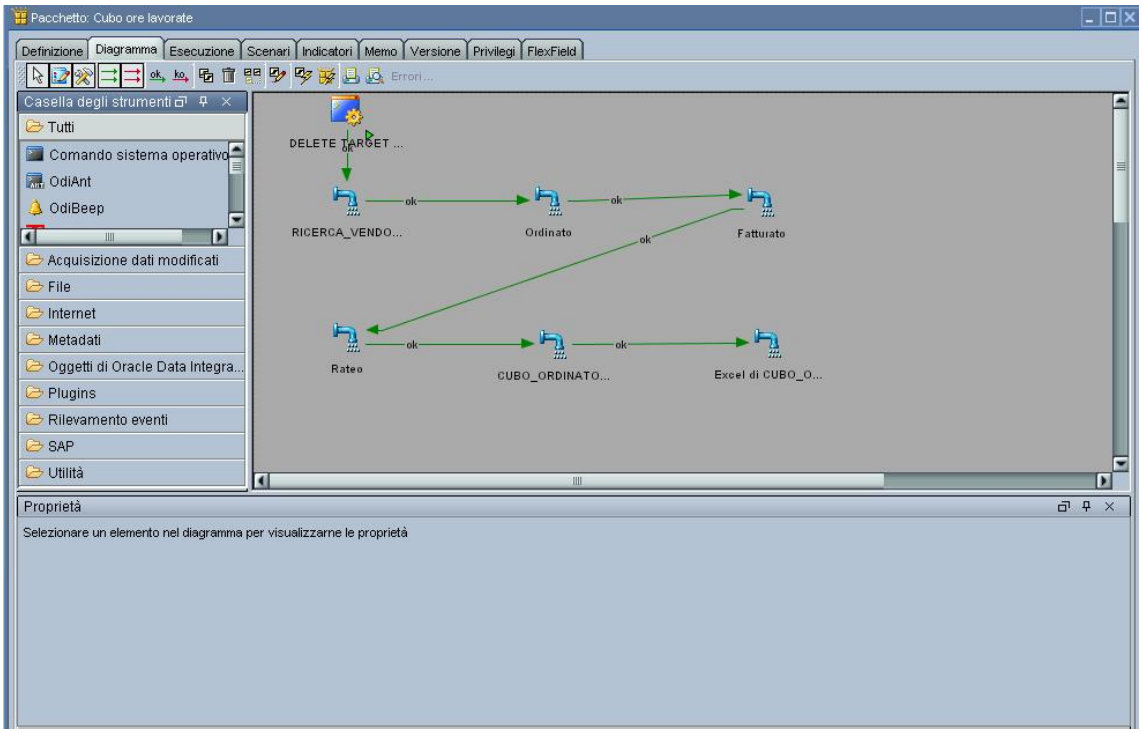


Figura 3.31: Diagramma del package Cubo Ore Lavorate

Innanzitutto bisognerà inserire all'interno del diagramma tutti quegli ele-

menti che si vuole che ODI lanci uno di seguito all'altro, dopodichè si dovrà stabilire l'ordine di esecuzione. Oltre alle interfacce che servono per il caricamento della tabella finale, viene inserita anche una procedura, creata in precedenza, che serve semplicemente a eliminare tutti i record delle tabelle target, poichè esse potrebbero già popolate. Tale procedura rappresenterà il punto di inizio del nostro workflow. A questo punto dovranno essere unite tutte le interfacce stabilendo l'ordine opportuno di esecuzione. Infine, collegata anche l'ultima interfaccia, il package sarà ultimato e pronto per essere eseguito. Dall'operator saremo in grado di seguire l'avvio del package, nello stesso modo in cui prima osservavamo il lancio delle singole interfacce.

Ultimato lo sviluppo del package, sarà adesso possibile avviarlo automaticamente in un ambiente di produzione, tramite la realizzazione di uno scenario. Per farlo sarà sufficiente cliccare col tasto destro l'icona del package appena creato e scegliere l'opzione che permette di generarlo. In tal modo sarà possibile lanciare automaticamente il package di interfacce in momenti prestabiliti gestendone la pianificazione. Inoltre il tutto potrà essere lanciato non soltanto nell'ambiente di ODI, ma anche tramite la shell del sistema operativo, indicando nome e versione dello scenario, da noi precedentemente definiti nella fase della sua creazione.

Capitolo 4

ETL CON PL/SQL

Nel precedente capitolo è stata effettuata l'implementazione dei flussi ETL per il caricamento dei cubi tramite l'utilizzo di Oracle Data Integrator. In questa parte della tesi, invece, la stessa cosa viene mediante programmi in PL/SQL; l'obiettivo sarà quello di spiegarne lo sviluppo, mostrando le parti più importanti del codice con cui sono stati realizzati. Tali programmi vengono chiamati "ERS Cubo Ore Lavorate (ORELAVOR)" e "ERS Cubo Acquisti (VEOLIA02)" e permettono, quindi, una volta avviati, di generare i file Excel utilizzati per caricare i cubi, popolando contemporaneamente le tabelle corrispondenti.

Già in questa fase si provvederà a confrontare similitudini e differenze di implementazione, rispetto a quanto già visto nel capitolo precedente, in modo da facilitare l'estrapolazione delle considerazioni utili per il confronto critico tra le due soluzioni adottate, che verranno proposte nel prossimo capitolo.

4.1 L'implementazione del cubo Ore Lavorate

Il primo estrattore che si esamina è quello che consente il caricamento del cubo delle ore lavorate. In questo caso, nella fase di partenza, si risparmia, nei confronti di ODI, la parte di configurazione del software, che comunque va effettuata solo per la prima volta, e quella di costruzione della topologia, necessaria, se si utilizza il software, per l'esecuzione dei progetti. Il listato 4.1 mostra la parte iniziale del codice del programma considerato, in cui si

Listato 4.1: Dichiarazione della procedura e lista parametri

```
1  PROCEDURE Ers_Dc_Crea_OreLavor
2                   (
3                    errcode         out  number,
4                    errbuff         out  varchar2,
5                    old_societa      in  varchar2,
6                    anno_estr        in  varchar2,
7                    mese_estr        in  varchar2
8                   )
9            is
```

inizia lo sviluppo della procedura, dichiarando una lista di parametri che essa richiede in input per poter essere lanciata, o che restituirà al termine della sua esecuzione.

I tre valori in input rappresentano il vecchio codice (il nuovo viene desunto a partire da esso in maniera automatica) che viene utilizzato per identificare un'azienda del gruppo Dalkia di cui estrarre i corrispondenti record, e l'anno e il mese in cui si esegue l'estrazione. I due parametri restituiti in output, invece, servono a informare l'utente circa lo stato di completamento delle operazioni effettuate durante l'esecuzione del programma, e saranno valorizzati in seguito. In ODI i problemi incontrati durante l'esecuzione delle interfacce sono gestiti automaticamente dal software, mentre, per quel che riguarda la lista di parametri da immettere, possono essere definite delle variabili, valorizzate dall'utente prima dell'esecuzione, che vengano utilizzate dalle interfacce per disporre delle informazioni necessarie al lancio.

A questo punto, iniziata la fase dichiarativa della procedura, vengono definiti e inizializzati due cursori, che serviranno a memorizzare ognuno un insieme di record; questi ultimi, in seguito, saranno processati uno alla volta e alcuni dei loro campi saranno migrati verso le due tabelle da popolare per il caricamento di questo cubo: DWBUDPR e DWPINTE.

Il primo cursore creato viene chiamato RICERCA_PROJECT. Parte del codice necessario per la sua implementazione viene descritto nel listato 4.2.

Esso richiede, per memorizzare le informazioni necessarie, due campi in input, che vengono calcolati a partire dai valori old_societa, anno_estr e mese_estr, introdotti in fase di avvio del concurrent dall'utente. La query utiliz-

Listato 4.2: Il cursore RICERCA_PROJECT

```
1  Cursor RICERCA_PROJECT      (p_org_id             in number,
2                               p_data_estrazione    in varchar2)
3
4                    is
5       select ppa.project_id                        id_commessa,
6              ppa.segment1                          codice_commessa,
7              ppa.name                              nome_commessa
8              (...)
9       from   PA_PROJECTS_ALL                ppa,
10             ERS_PA_ORG_HIERARCHY_V         vpr
11      where  ppa.org_id                      =      p_org_id
12             and substr(ppa.segment1,1,1)    in     ('M', 'W')
13             and ppa.project_id              =      vpr.project_id
14             (...)
```

zata per popolarlo va esattamente a trattare tabelle sorgenti, data mapping e filtri descritti nella parte di progettazione funzionale relativa alla realizzazione dell'omonima tabella. Al contrario di quanto visto in ODI, quindi, questa volta, per mantenere i record da processare nella seconda parte del flusso e per i quali calcolare i valori aggregati, è possibile utilizzare dei cursori, senza far ricorso alla memorizzazione fisica dei record in tabelle. Nello specifico, RICERCA_PROJECT servirà a popolare direttamente la tabella DWBUDPR, collegando i record in esso mantenuti con altre tabelle, mentre i suoi record verranno letti singolarmente dal secondo cursore, dichiarato immediatamente dopo, per calcolare per ognuno di essi i valori aggregati delle ore lavorate consuntive. Parte dell'implementazione di quest'ultimo, chiamato "ESTRAE_ORELAVOR_CONS", è visualizzata nel listato 4.3.

Il cursore, ogni volta che viene letto, richiede in input, oltre al parametro relativo alla società trattata, il valore di un campo dei record presenti in RICERCA_PROJECT, precisamente quello relativo al codice della commessa. Nella parte del corpo del programma, l'utilizzo di un ciclo permetterà di poter far processare al cursore, ad ogni iterazione, l'i-esimo codice commessa presente dei record di RICERCA_PROJECT, per recuperare tutte le righe collegate a quella commessa che descrivono i dettagli relativi alle ore lavorate consuntive della stessa.

La query che viene utilizzata per la sua implementazione, in questo caso, riprende le business rules che servivano a limitare le ore lavorate da consi-

Listato 4.3: Il cursore ESTRAE_ORELAVOR_CONS

```
1  Cursor ESTRAE_ORELAVOR_CONS   (p_project_id         in number,
2                                 p_data_estrazione    in varchar2)
3                  is
4      select nvl(pei.quantity,0)             peia_quantity,
5             nvl(pei.burden_cost,0)          peia_costo,
6             null                            peia_prestazione,
7             null                            peia_attivita,
8             null                            peia_squadra,
9             pap.employee_number             employee_number
10            (...)
11     from   PA_EXPENDITURE_ITEMS_ALL       pei,
12            PA_TASKS                       pta,
13            ERS_CDG_PA_PROJECT_TASK_V      ppt,
14            PA_EXPENDITURES_ALL            pal,
15            PER_ALL_PEOPLE_F               pap
16     where  nvl(pei.project_id,-2) = p_project_id  -- join con la
           commessa contenuta nel cursore RICERCA_PROJECT
17            (...)
```

derare per il calcolo dei consuntivi in base a determinati requisiti del cliente (anche in questo caso si rimanda al capitolo della progettazione funzionale per maggiori dettagli). I campi che vengono memorizzati sono quelli che ritroveremo all'interno della tabella di destinazione DWPINTE, che tale cursore serve a popolare. Fra questi, alcuni riguarderanno le dimensioni trattate per il cubo delle ore lavorate e i campi che rappresentano gli aggregati richiesti.

Sempre all'interno della parte dichiarativa della procedura, vengono inizializzate diverse variabili, che possono essere descritte a gruppi, a seconda dello scopo che esse avranno nella parte successiva dello sviluppo del concurrent. Il primo gruppo di campi che viene dichiarato è quello che servirà per memorizzare l'intero path in cui saranno appoggiati i file csv che manterranno le informazioni estratte e che verranno in seguito importati con Excel per la realizzazione dei prospetti. Altri parametri, invece, saranno utili come flag per segnalare quando i curosri vengono utilizzati per la prima volta, poichè in questo caso sarà necessario effettuare altri tipi di operazioni. Il listato 4.4 mostra l'inizializzazione delle variabili citate.

Raffrontando l'implementazione appena descritta con quanto fatto attraverso l'utilizzo di ODI, si nota che tutte queste azioni, ed in particolar modo la gestione dei file in cui migrare i dati che fanno parte dell'indagine, sono facilitate dall'utilizzo dei moduli grafici e degli knowledge module, in seguito

Listato 4.4: Dichiarazione campi per la memorizzazione dei path

```
1  out_dir_name                  varchar2(200);
2  out_file_dwbudpr              varchar2(200);
3  l_handle_dwbudpr              UTL_FILE.FILE_TYPE;
4  out_file_dwpinte              varchar2(200);
5  l_handle_dwpinte              UTL_FILE.FILE_TYPE;
6  code_err                      number              := null;
7  desc_err                      varchar2(2000)      := null;
8  v_prima_dwbudpr               varchar2(2)         := 'SI';
9  v_prima_dwpinte               varchar2(2)         := 'SI';
```

Listato 4.5: Dichiarazione campi di controllo

```
1  v_failure                     varchar2(2)         := 'NO';
2  v_errori                      varchar2(2)         := 'NO';
3  v_errors                      varchar2(250)       := null;
4  v_trattare                    varchar2(2)         := 'NO';
5  l_seq_lancio                  number              := null;
6  v_user_id                     number              := null;
7  v_sob_id                      number              := null;
8  v_org_id                      number              := null;
9  v_org_code                    varchar2(3)         := null;
10 (...)
```

alla realizzazione di una topologia adatta agli scopi che ci si prefigge.

Nel listato 4.5 vengono inizializzati o soltanto dichiarati dei campi di lavoro che saranno utilizzati nelle parti successive dello sviluppo. Fra essi alcuni saranno utili per il controllo degli errori, che deve essere gestito in questo caso dai programmatori, per avvertire l'utente circa la presenza di problemi e il conseguente fallimento del lancio.

Infine, nella parte dichiarativa della procedura, compaiono quei contatori che ospiteranno i valori aggregati relativi alle ore lavorate e improduttive. In pratica, come si vedrà dettagliatamente in seguito, questi valori, utilizzati all'interno di un ciclo, serviranno per effettuare le sommatorie delle quantità numeriche richieste per quegli insiemi di record che vengono restituiti raggruppando in base a più valori discreti. I dettagli relativi all'inizializzazione di questi campi viene mostrata nel listato 4.6.

Le variabili mostrate nel codice serviranno per effettuare la memorizzazione dei record all'interno della tabella (quelli che iniziano per "v") e degli stessi valori all'interno del file csv (quelli che iniziano per "x"). In ODI la

Listato 4.6: Dichiarazione delle variabili aggregate

```
1  v_val_budget                number              := 0;
2  v_ore_budget                number              := 0;
3  v_ore_ordinarie             number              := 0;
4  v_ore_improduttive          number              := 0;
5  v_val_ordinarie             number              := 0;
6  v_val_improduttive          number              := 0;
7  x_val_budget                number              := 0;
8  x_ore_budget                number              := 0;
9  x_ore_ordinarie             number              := 0;
10 x_ore_improduttive          number              := 0;
11 x_val_ordinarie             number              := 0;
12 x_val_improduttive          number              := 0;
```

Listato 4.7: Creazione file di log

```
1 begin
2 DBMS_OUTPUT.ENABLE(990000);
3 v_data_log    :=  TO_CHAR(sysdate,'DD/MM/YYYY hh24:mi:ss');
4 v_proc_log    :=  ' - Ers_Dc_Crea_OreLavor - ';
5 ToLog(v_data_log ||' Inizio Procedura' || v_proc_log);
6 out_dir_name      := '/tmp/svildb/10.2.0/admin/SVIL_demodb/tmp';
```

migrazione delle informazioni verso la tabella piuttosto che verso il file csv, viene gestita specificando in maniera opportuna i knowledge module.

Terminata la parte dichiarativa, l'istruzione BEGIN preannuncia l'inizio del corpo della procedura. La prima parte, in particolare, serve per la creazione dei file di log che permetteranno di esaminare i dettagli relativi all'esecuzione del concurrent e saranno utili, nello specifico, per le fasi di debugging e di testing. Abbiamo in precedenza visto che in ODI, invece, log e statistiche dettagliate vengono mostrate dal modulo grafico Operator. Nel primo listato che viene proposto (il 4.7) appaiono i comandi necessari per l'implementazione dell'intestazione dei file di log che comprende la registrazione della data da memorizzare in essi, che si riferisce al giorno in cui è stato eseguito il programma (e quindi si è effettuata l'estrazione delle informazioni) e il nome della procedura che è stata lanciata.

Nella parte successiva di codice, vengono richiamare delle procedure esterne, create in precedenza, per effettuare una serie di operazioni, importanti per una corretta esecuzione del concurrent. La prima di queste è la ricerca dell'utente cui spetterà il lancio del programma, e quindi la fase di

integrazione dei dati. Nel listato 4.8 viene mostrata la parte riferita ad essa.

Listato 4.8: Ricerca dell'utente di data integration

```
1  begin
2      select user_id  into  v_user_id
3           from FND_USER            fnu
4              where user_name          =  'DATA CONVERSION'
5                and trunc(sysdate)  between  trunc(nvl(start_date,
                     sysdate))  and  trunc(nvl(end_date,sysdate));
6                  exception when no_data_found then v_failure := 'SI';
7                               ToLog(v_data_log||v_proc_log||' User
                                    assente su "FND_USER" '||'DATA
                                    CONVERSION');
8                            when others        then v_failure := 'SI';
9                               ToLog(v_data_log||v_proc_log||' errore
                                    in fase di controllo dello "User"
                                    '||'DATA CONVERSION '||substrb(
                                    SQLERRM,1,100));
10 end;
```

Tramite questi comandi, sarà memorizzato in una variabile l'utente della tabella FND_USER che ha il campo user name impostato a "Data Conversion", e che è valido rispetto alla data in cui si effettua l'estrazione. In seguito al successo o fallimento di questa operazione, verranno conseguentemente aggiornati i log.

Un'altra operazione importante da effettuare in questa fase è la conversione del parametro richiesto all'utente old_societa nella variabile v_org_id, che sarà utilizzata in seguito; ciò viene fatto richiamando una apposita procedura, memorizzando poi il risultato dell'operazione nei log. La stessa cosa viene fatta per la conversione dei parametri di input anno_estr e mese_estr nella variabile v_data_estr, con conseguente aggiornamento dei log. Infine, nella parte di listato visualizzato, vengono riempiti dei campi con il valore relativo al nome dei file Excel da creare e memorizzare nel percorso definito in precedenza.

Terminate anche queste operazioni, si passa ad esaminare la parte di codice in cui vengono implementate le business rules e in cui vengono sfruttati i due cursori, tramite l'utilizzo di due cicli precedentemente dichiarati, per calcolare i valori aggregati di interesse. Nel listato 4.9 viene visualizzata la parte di implementazione relativa all'inizio dell'esecuzione del primo ciclo.

Listato 4.9: Parte del ciclo esterno

```
1   begin
2       for pro in RICERCA_PROJECT     (v_org_id,  v_data_estr)
3         loop
4            ToLog('Sto trattando commessa => '||pro.codice_commessa
                 ||' con "ID" => '||pro.id_commessa);
5            v_trattare              := 'NO';
6            v_ore_budget            := 0;
7            v_val_budget            := 0;
8            begin
9                select sum(nvl(bdg.bdg_quantity,0)),
10                      sum(nvl(bdg.bdg_raw_cost,0))
11                            into
12                      v_ore_budget,
13                      v_val_budget
14               from   ERS_CDG_PA_PROJECT_BUDGET_V    bdg,
15                      PA_BUDGET_VERSIONS             pbv,
16                      PA_FIN_PLAN_TYPES_B            fpt
17               where  (...)
18                    if   v_ore_budget  !=  0              or
19                         v_val_budget  !=  0
20                    then
21                         v_trattare  := 'SI';
22                    end if;
```

Il loop permette di trattare il contenuto del cursore RICERCA_PROJECT un record alla volta, ovvero una commessa per volta. Per ognuna di esse, effettuando dei collegamenti con le tabelle relative ai dati dei budget, rispettando le restrizioni citate nel capitolo dellla progettazione funzionale riguardanti i tipi di budget da considerare, verranno restituiti più record, tramite cui, sommando i valori relativi alle ore previste e al corrispondente valore monetario, potranno essere valorizzati i due campi v_ore_budget e v_val_budget, che conterranno il tal modo gli importi richiesti. Nei log verrà scritto l'id della commessa che si sta trattando in quella determinata iterazione del loop. Un ulteriore controllo permetterà di inserire nella tabella DWBUDPR, relativa alle ore complessive previste nel budget per ogni commessa, e nel corrispondente file Excel, solo quelle commesse per cui i valori calcolati sono di interesse per l'indagine, quindi diversi da zero.

Successivamente è prevista l'esecuzione di un secondo loop interno (vedi listato 4.10), in cui saranno questa volta utilizzati i record contenuti nel cursore ESTRAE_ORELAVOR_CONS. In particolare verranno recuperati i record delle ore lavorate memorizzate che si riferiscono alla i-esima commessa presa

Listato 4.10: Parte del ciclo interno

```
1  begin
2            for csn in ESTRAE_ORELAVOR_CONS  (pro.id_commessa,
                 v_data_estr)
3            loop
4                v_trattare           := 'SI';
5                v_ore_ordinarie      := 0;
6                v_ore_improduttive   := 0;
7                v_val_ordinarie      := 0;
8                v_val_improduttive   := 0;
9                if lpad(nvl(csn.peia_attivita,'0'),2,'0')  <  '51'
10               then
11                   v_ore_ordinarie      := nvl(csn.peia_quantity,0);
12                   v_val_ordinarie      := nvl(csn.peia_costo,0);
13               else
14                   v_ore_improduttive   := nvl(csn.peia_quantity,0);
15                   v_val_improduttive   := nvl(csn.peia_costo,0);
16               end if;
```

in input dal cursore precedente e che si sta trattando al momento. Le righe recuperate verranno trattate singolarmente, una per ogni iterazione di questo secondo ciclo interno.

Considerando ad esempio il primo record, ciò che verrà fatto sarà popolare i campi delle ore ordinarie e del corrispettivo valore monetario (v_ore_ordinarie e v_val_ordinarie) effettuando dei join tra il suo campo codice commessa e i campi di join di altre tabelle sorgenti e memorizzando in essi i valori che corrispondono alle ore lavorate e ai relativi importi monetari, raggruppando per tutte le dimensioni da considerare. Come si può vedere, viene implementato un controllo basato sul valore del campo peia_attivita che servirebbe a distinguere i casi in cui popolare i valori delle ore ordinarie da quelli in cui popolare le ore improduttive. Poichè però questo campo è sempre privo di valori (e quindi uguale a zero per la funzione nvl), verranno migrate solo le ore ordinarie, mentre per quelle improduttive verrà creato lo spazio ma i loro campi non saranno mai popolati.

A questo punto il record della i-esima iterazione del ciclo interno è pronto per essere scritto in DWPINTE, sia nella tabella che nel corrispondente file Excel, definito in precedenza. Il listato 4.11 mostra parte della sua trascrizione, campo per campo, nella tabella di destinazione, definendo la sorgente da cui recuperare le informazioni per ogni colonna da valorizzare. Alcune di que-

Listato 4.11: Caricamento dati del record nella tabella

```
1  begin
2            insert into DWPINTE
3                  (
4                  stagione,
5                  anno_rapportino,
6                  mese_rapportino,
7                  codice_risorsa,
8                  regione_commessa,
9                  (...)
10                 )
11           values
12                 (
13                 csn.ppt_stagione,
14                 anno_estr,
15                 mese_estr,
16                 csn.employee_number,
17                 pro.region_name,
18                 (...)
19                 );
20           exception when others then v_failure:= 'SI';
21           ToLog(v_data_log||v_proc_log||' Errore in fase di
                 scrittura di           "DWPINTE" ||pro.codice_commessa
                 ||' '||pro.sector_name||' '||csn.pta_task_number||'
                 '||csn.employee_number||' '||substrb(SQLERRM,1,100));
22                 end;
```

ste, come già visto per l'implementazione in ODI, vengono mantenute vuote. I valori corrispondenti verranno recuperati nelle successivi fasi di integrazione, prima del caricamento del cubo.

Terminata la scrittura del record nella tabella, si inizia il trasferimento delle informazioni nel file di formato csv. La prima cosa che va fatta in questo caso è scrivere dei comandi per aprire il file in cui inserire i valori. Ciò deve essere implementato dal programmatore gestendo, tramite opportuni comandi, la fase di apertura, scrittura e chiusura del file, e informando l'utente delle operazioni svolte durante l'esecuzione della procedura nei log che si stanno scrivendo; nel caso di ODI, invece, il software, automaticamente, indicando il knowledge module opèportuno andava ad eseguire tutte le operazioni necessarie.

A questo punto sarà possibile migrare all'interno del file i dati recuperati in precedenza. Il listato 4.12 mostra i passi eseguiti per realizzare questa fase dell'implementazione.

In esso vediamo che gli aggregati, che sono stati valorizzati in precedenza

Listato 4.12: Caricamento dati del record nel file csv

```
x_ore_ordinarie    := Ers_Dc_Gest_Info_Pk.Ers_Dc_Norm_Numeri(
    v_ore_ordinarie, 5, 2);
                      x_val_ordinarie    := Ers_Dc_Gest_Info_Pk.
                          Ers_Dc_Norm_Numeri(v_val_ordinarie, 17, 5);
                      x_ore_improduttive := Ers_Dc_Gest_Info_Pk.
                          Ers_Dc_Norm_Numeri(v_ore_improduttive, 5,
                          2);
                      x_val_improduttive := Ers_Dc_Gest_Info_Pk.
                          Ers_Dc_Norm_Numeri(v_val_improduttive, 17,
                          5);
UTL_FILE.PUT_LINE
       (
       l_handle_dwpinte,
       csn.ppt_stagione    ||
       chr(09)             ||
       anno_estr           ||
       chr(09)             ||
       mese_estr           ||
       chr(09)             ||
       csn.employee_number ||
       chr(09)             ||
       pro.region_name
       (...)
       );
end loop
```

e scritti nella tabella, vengono normalizzati e memorizzati in nuove variabili tramite una procedura appositamente realizzata, affinchè essi possano essere inclusi senza problemi in un file csv e poi letti da Excel. In seguito il record della i-esima iterazione viene finalmente scritto nel file, campo per campo e aggiungendo di seguito la sua intestazione. Nel listato vengono visualizzati i dettagli relativi solo ai primi cinque campi, a titolo di esempio.

Eseguita anche quest'ultima operazione, il ciclo interno si chiude, e di seguito si procede alla memorizzazione del record relativo alla i-esima iterazione del ciclo più esterno, che si riferisce ai record del cursore RICERCA_PROJECT e che viene questa volta memorizzato in DWBUDPR, poichè quest'ultima tabella si riferisce alle informazioni sulle ore lavorate previste nel budget per le varie commesse. Le stesse operazioni viste in precedenza, dalla scrittura nella tabella all'apertura del file e alla scrittura in esso, vengono in questo caso ripetute ed esattamente nello stesso ordine.

Infine, per quel che riguarda l'esecuzione anche di questo ciclo più esterno, viene eseguito il COMMIT della transazione (o il ROLLBACK se è stata

Listato 4.13: Operazioni di controllo e fine della transazione SQL

```
1  if v_failure = 'SI'
2                 then
3                 v_errori  := 'SI';
4                 v_err_ore_budget        := v_err_ore_budget        +
                       nvl(v_ore_budget,0);
5                 v_err_val_budget        := v_err_val_budget        +
                       nvl(v_val_budget,0);
6                 (...)
7                 ROLLBACK;
8  else
9                 v_tot_ore_budget        := v_tot_ore_budget        +
                       nvl(v_ore_budget,0);
10                v_tot_val_budget        := v_tot_val_budget        +
                       nvl(v_val_budget,0);
11                (...)
12                COMMIT;
13 end if;
14 v_failure := 'NO';
15 end loop; -- fine del ciclo esterno
```

rilevata una condizione di errore od eccezione) e vengono effettuate delle operazioni di controllo. Nello specifico, se saranno stati incontrati degli errori durante l'esecuzione dei due cicli, verranno progressivamente popolati i campi relativi ai valori di errore, altrimenti in altre variabili di controllo saranno incrementalmente sommate le ore ordinarie contate durante le varie iterazioni dei cicli (i campi relativi erano stati creati e inizializzati in precedenza), come viene mostrato nel listato 4.13.

Queste informazioni, comunque, non saranno mai migrate verso alcuna tabella, ma soltano trascritte nei file di log, di modo che, una volta terminate tutte le iterazioni, si abbiano informazioni riguardo il totale delle ore migrate correttamente, sia per le ore ordinarie, che per quelle improduttive. Anche queste operazioni vengono gestite automaticamente da ODI tramitei suoi moduli grafici che riportano dati relativi ai record correttamente trasferiti, mentre, in questo caso, devono essere sviluppate dai programmatori.

A questo punto, nello sviluppo della procedura, è presente il comando end loop, che porta nuovamente l'esecuzione al punto iniziale del ciclo esterno, rieffettuando per ogni record presente nel cursore RICERCA_PROJECT tutte le operazioni già descritte, fin quando non saranno stati considerate tutte le righe, e le due tabelle DWBUDPR e DWPINTE non saranno state popolate interamente. Solo a questo punto, si uscirà dall'esecuzione di entrambi i cicli

e ciò che verrà fatto, a completamento della procedura di creazione del cubo relativo alle ore lavorate, sarà chiudere i file csv creati, importabili da Excel, ed effettuare le ultime operazioni di scrittura dei log per informare dell'avvenuta esecuzione senza errori della procedura o, al contrario, per avvertire che sono stati riscontrati dei problemi effettuando qualcuna delle operazioni vista finora.

Come abbiamo visto, quindi, i programmi sviluppati in PL/SQL sfruttano una logica che permette, tramite l'utilizzo dei cursori, l'estrazione e la scrittura dei dati record a record, rendendo inutile la memorizzazione dei flussi intermedi in apposite tabelle; al contrario, in ODI, si sono dovuti realizzare degli archivi temporanei in cui memorizzare soltanto i record per i quali realizzare le tabelle che servono per il caricamento del cubo.

4.2 L'implementazione del cubo Ordini Materiali

Terminata l'implementazione del flusso che permette il caricamento delle tabelle atte a popolare il cubo delle ore lavorate, si passa adesso ad esaminare il processo ETL relativo al cubo degli ordini richiesti ai venditori da Siram. La tabella da creare in questo caso è quella in precedenza chiamata DWFATVE2. Diverse operazioni da effettuare sono del tutto simili a quelle viste nel paragrafo precedente e, di conseguenza, una buona parte di codice verrà soltanto citata e non visualizzata tramite i listati.

Nella parte dichiarativa della procedura, anche in questo caso, vengono inizializzate tutte quelle variabili e quei campi di lavoro che saranno utilizzati in seguito. Fra essi troviamo i campi numerici, che conterranno i valori aggregati da considerare (le misure di questo cubo) e da migrare verso i supporti di destinazione, le variabili relative alla memorizzazione degli stessi campi nel file csv, da importare in Excel, e infine i campi che serviranno per effettuare le quadrature finali dei conti, memorizzate nei log a scopo di controllo.

Per il cubo degli ordini, il cursore che viene dichiarato in questa fase, e in seguito utilizzato, è uno solo, e viene come al solito chiamato RICERCA_VENDOR_CONTO. Parte della sua implementazione è mostrata nel listato 4.14.

Listato 4.14: Il cursore RICERCA_VENDOR_CONTO

```
1  Cursor RICERCA_VENDOR_CONTO      (p_org_id             in number,
2                                    p_data_estr_iniz     in date,
3                                    p_data_estr_fine     in date)
4                  is
5       select pha.vendor_id                                 vendor_id,
6              pve.segment1
                   codice_fornitore,
7              substr(nvl(pvs.attribute4,'??'),1,2)
                   divisione_fornitore,
8              pda.expenditure_type
                   expenditure_type,
9              cat.category_id                               categoria,
10             (...)
11      from   PO_HEADERS_ALL          pha,
12             PO_VENDORS              pve,
13             PO_VENDOR_SITES_ALL     pvs,
14             (...)
15      where  pha.org_id = p_org_id
16      and    pha.type_lookup_code = 'STANDARD'
```

I record che vengono mantenuti in questo cursore sono quelli relativi ai soli tipi di ordini per cui gli tulizzatori finali del cubo vogliono compiere le indagini descritte nei requisiti e per i quali si vuole che vengano calcolati gli aggregati di interesse. Si rimanda, ancora una volta, al capitolo delle progettazione funzionale per la descrizione delle business rules che vengono imposte per raggiungere il risultato desiderato. I record che rispettano queste restrinzioni vengono raggruppati rispetto alle dimensioni da considerare per il cubo, ovvero divisione, categoria, conto e fornitore (il group by non viene mostrato nel listato).

In seguito, nella parte iniziale del corpo della procedura, vengono ripetute, nello stesso identico modo, tutte quelle operazioni incontrate, nello stesso punto, nel precedente paragrafo per il cubo delle ore lavorate: creazione dei file di log, memorizzazione dei percorsi in cui mantenere i file csv che verranno creati, ricerca dell'utente cui competerà la data migration e conversione dei parametri richiesti in input dai cursori a partire dalle variabili immesse dall'utente all'avvio della procedura.

A questo punto, vengono implementati loop e operazioni all'interno di esso che consentono di trattare singolarmente ogni record presente nel cursore dichiarato in precedenza, in modo da, per ognuno di essi, calcolare i campi aggregati richiesti. Inoltre, sempre all'interno del ciclo, vengono popolati,

Listato 4.15: Valorizzazione categoria e conto

```
1  begin
2         for rvc in RICERCA_VENDOR_CONTO  (v_org_id,  v_data_estr_iniz
              ,  v_data_estr_fine)
3           loop
4            v_categoria     := rvc.segment1_category||rvc.
                segment2_category||rvc.segment3_category||rvc.
                segment4_category;
5
6  begin
7                  select ltrim(rtrim(substr(description,1,instr(
                       description, ' ')-1)))  into rvc.conto
8                        from PA_EXPENDITURE_TYPES
9                           where expenditure_type
                                                          =  rvc.
                                 expenditure_type
10 and to_date(v_data_estr||'01','YYYYMMDD')        between
       start_date_active  and  nvl(end_date_active,to_date(v_data_estr
       ||'01','YYYYMMDD')) and rownum =  1;
11 end;
```

relativamente all'ordine di acquisto trattato, il campo della categoria (come concatenazione dei quattro campi che descrivono il tipo di acquisto effettuato) e quello del conto, che nel cursore era stato dichiarato ma lasciato vuoto. La loro implementazione è visualizzata nel listato 4.15.

Per ogni record presente nel cursore, tre query distinte permettono di valorizzare i tre aggregati da visualizzare nei risultati: importo_ordinato, importo_rateo e importo_fatturato. Ognuna di queste effettua collegamenti con tabelle diverse e impone regole di restrinzione differenti, in modo da calcolare opportunamente, seguendo i requisiti del cliente, le somme numeriche. Di seguito, nel listato 4.16, viene visualizzata la parte relativa al calcolo dell'importo ordinato per gli ordini da considerare.

Come si può notare dal codice, vengono sfruttati i campi relativi alle dimensioni, per cui in precedenza si sono raggruppati i dati, per collegarsi alle tabelle che permettono di ricavare le informazioni relative all'importo ordinato (le uniche condizioni visualizzate dopo la parola chiave WHERE sono proprio i join citati). Nel capitolo della progettazione funzionale possono essere reperiti maggiori dettagli riguardo le business rules da definire, le tabelle sorgenti da cui prelevare i dati necessari e le modifiche da effettuare a quanto visto in precedenza per calcolare anche gli importi del fatturato e del rateo,

Listato 4.16: Calcolo degli aggregati relativi all'importo ordinato

```
1  begin
2       select sum(pll.quantity  -  pll.quantity_cancelled -  pll.
            quantity_rejected),
3              sum((pll.quantity  -  pll.quantity_cancelled -  pll.
                   quantity_rejected) * pla.unit_price)
4       into
5              v_qta_ordinata,
6              v_val_ordinato
7       from   PO_HEADERS_ALL          pha,
8              PO_LINES_ALL            pla,
9              PO_DISTRIBUTIONS_ALL    pda,
10             (...)
11      where  pha.vendor_id =  rvc.vendor_id
12      and    nvl(pvs.attribute4,'?') = rvc.divisione_fornitore
13      and    nvl(pda.expenditure_type,'??') = rvc.expenditure_type
14      and    pla.category_id = rvc.categoria
15             (...)
```

Listato 4.17: Valorizzazione dell'importo del rateo

```
1  if v_val_fatturato  != 0
2                 then
3                 v_val_rateo  :=  v_val_rateo  -  v_val_fatturato;
4                 if v_val_rateo  <  0
5                    then
6                    v_val_rateo  :=  0;
7                 end if;
8              end if;
```

rispetto alle dimensioni definite.

Una volta calcolato il valore di tutti gli aggregati per l'ordine in questione, come si vede nel listato 4.17, è necessario che venga trattato nuovamente il campo del rateo, a seconda di ciò che compare, in relazione allo stesso ordine, nel valore del fatturato, come già visto nel secondo capitolo.

A questo punto, nella parte finale del ciclo, tutte le informazioni che sono state memorizzate per il record che si sta trattando, vengono memorizzate nella tabella DWFATVE2 e nel file csv corrispondente. Le informazioni relative all'ordine che si vogliono mantenere, come visto anche nei precedenti capitoli, sono: codice della società che richiede gli ordini, divisione dell'azienda che chiede la fornitura, categoria e conto dell'acquisto, codice del fornitore, un booleano per stabilire se il fornitore è convenzionato, gli importi dell'ordinato, fatturato e rateo e le informazioni relative al periodo in cui si è effettuata

l'estrazione.

Prima di caricare i dati, però, viene imposto un controllo aggiuntivo che consente di non migrare verso la tabella e il file csv quei record per cui contemporaneamente accade che tutti e tre i valori aggregati richiesti siano uguali a zero; in questo caso l'ordine in questione va scartato.

Infine vengono sostanzialmente ripetute tutte le operazioni già descritte in dettaglio per l'implementazione del cubo precedente. Tramite comandi SQL, i valori del record attualmente trattato e da migrare vengono memorizzati nella tabella DWFATVE2, e, dopo aver normalizzato i valori aggregati ed aperto il file csv, riversati anche in quest'ultimo.

Queste operazioni segnalano la fine del ciclo, iniziato con la lettura dei dati del cursore. Di conseguenza, se ancora sono presenti ordini da trattare in RICERCA_VENDOR_CONTO, vengono ripetute tutte le fasi viste in precedenza, per ognuno di essi, fino alla scrittura dei loro campi nella tabella e nel file; altrimenti si esce dal ciclo e vengono effettuate le ultime operazioni, prima della fine della procedura. Esse consistono nell'esecuzione della transazione SQL (tramite COMMIT), nell'aggiornamento dei log e nella chiusura del file csv, che va effettuata obbligatoriamente dopo aver in esso riversato tutte le informazioni da migrare.

Capitolo 5

BENCHMARK: PARAGONE TECNICO

Nei precedenti capitoli è stata analizzata l'implementazione dei flussi ETL per il caricamento dei cubi prima utilizzando il tool Oracle Data Integrator e poi sviluppando dei programmi in linguaggio PL/SQL. In questo e nel prossimo capitolo l'obiettivo sarà quello di paragonare le due soluzioni adottate, in maniera da valutare pregi e difetti per ognuna di esse. Nello specifico, in questa parte sarà effettuato un confronto critico relativamente agli aspetti tecnologico-implementativi, basato su diversi parametri di valutazione qualitativi e quantitativi, presentati al momento di eseguire il raffronto.

L'analisi sarà divisa in tre fasi distinte poichè proprio in tre parti è stato scomposto l'intero processo di ciclo di vita del software oggetto dell'analisi, e poichè è necessario disporre di considerazioni finali per ognuno di questi step per avere un quadro più chiaro del confronto. Verranno presi in considerazione più indicatori per ognuna delle fasi, ovvero analisi di dettaglio e sviluppo dei flussi ETL, rilascio in produzione e manutenzione correttiva-evolutiva degli estrattori sviluppati. I risultati dell'analisi saranno sintetizzati, alla fine di ogni paragrafo, in dei grafici che mettano in risalto in maniera intuitiva le differenze fra le due teniche utilizzate per ognuno dei termini di paragone.

5.1 Analisi di dettaglio e sviluppo

La prima fase che costituisce un progetto di implementazione di flussi ETL consiste nell'analisi e nell'implementazione degli estrattori che consentono di migrare i record trasformati nelle tabelle di destinazione definite. Di seguito si procede alla descrizione degli indicatori definiti e valutati per questo step, cui succede, per ognuno, un confronto dettagliato tra Oracle Data Integrator e lo sviluppo dei programmi in PL/SQL.

5.1.1 Facilità di integrazione di tipologie di sorgenti dati eterogenee

- **Descrizione**

  Data l'etereogenità di sorgenti di informazioni attualmente esistenti, risulta fondamentale la possibilità di integrarsi in maniera il più possibile trasparente con la maggior parte di esse. Il parametro preso di volta in volta in considerazione propone il confronto tra ODI e il linguaggio PL/SQL nei vari aspetti delle problematiche di integrazione; di seguito, saranno illustrate le soluzioni tecniche messe a disposizione da ciascuno degli strumenti considerati per la risoluzione di questo importante problema.

- **Confronto**

  Una delle caratteristiche più importanti di ODI è proprio quella di permettere facilmente la connessione e il caricamento delle informazioni estratte e trasformate da e verso diversi tipi di sorgenti informative, dai database di diverse tecnologie ai file di diversi tipi di formato. Ciò è reso possibile grazie all'esistenza degli knowledge modules, che definiscono le operazioni da effettuare per le fasi di lettura, integrazione e caricamento dei dati nel supporto di destinazione. Ogni knowledge module, quindi, risulta implementato in maniera diversa a seconda della tecnologia cui si deve connettere, e può essere selezionato a piacimento dall'utente per soddisfare qualsiasi esigenza di integrazione.

  Se ODI offre la possibilità di integrarsi trasparentemente con numerosi tipi di sistemi e permettere connessioni a sorgenti di dati multi-

ple, il PL/SQL, invece, non possiede nativamente queste caratteristiche, di conseguenza queste funzionalità, se necessarie, devono essere implementate direttamente dal programmatore, oppure devono essere supportate da moduli software specifici. Questo si traduce in un aumento di tempo richiesto per l'implementazione dei flussi ETL e di impegno e competenze richieste alle risorse che si occupando di questo lavoro.

Ad esempio, facendo riferimento ai flussi ETL realizzati nei capitoli precedenti, nell'uno e nell'altro modo, si nota che in ODI la connessione a tecnologie diverse comporta alcune modifiche effettuabili da un utente poco esperto, mentre dai listati PL/SQL si è potuto constatare in particolare che la migrazione dei dati verso i file txt ha comportato la scrittura di numerose linee di codice e l'utilizzo di procedure esterne per scopi di normalizzazione dei dati. Ipotizzando che, in un dato momento, i tipi di requisiti richiedano la connessione ad una nuova tecnologia, si può concludere che certamente questo cambiamento comporterebbe maggiori problemi per i programmatori PL/SQL che per gli sviluppatori di ODI.

Quanto si è detto porta a considerare l'integrazione con le sorgenti informative come un'operazione che possa occorrere anche in caso di modifiche evolutive del software; ciò nonostante, per ragioni di sintesi, questo parametro di valutazione è stato collocato nella fase di disegno e implementazione dei flussi.

5.1.2 Facilità di analisi ed implementazione degli estrattori dei dati

- **Descrizione**

  Dopo il disegno dei flussi ETL, la fase successiva ha come obiettivo l'analisi di dettaglio e lo sviluppo degli estrattori, cosi come sono stati progettati precedentemente. Questo si traduce inevitabilmente in un confronto sulla quantità di tempo necessaria per l'esecuzione di questo step. Il confronto da effettuare, quindi, propone il paragone tra le soluzioni tecniche offerte dai due strumenti utilizzati, facilitato dall'applicazione di entrambe ad un caso reale, come visto nei precedenti capitoli.

- **Confronto**

  Utilizzando il software, l'analisi e l'implementazione degli estrattori è di certo facilitata dalla possibilità di definire tabelle sorgenti, di destinazione e business rules utilizzando un supporto grafico, in seguito alla definizione della topologia del progetto, e dei modelli per ciscuno degli archivi da leggere. In pratica, l'utente può definire le operazioni da effettuare utilizzando il tool piuttosto che scrivendo righe di codice, come si è obbligati a fare nel caso della programmazione in PL/SQL. In quest'ultimo caso, per l'estrazione dei dati, è necessaria la mappatura manuale delle informazioni; ciò porta ad avere maggiori probabilità che siano commessi degli errori per l'esecuzione delle operazioni necessarie. Seppur la programmazione abbia come vantaggio la possibilità di riutilizzo del codice, non sembra sbagliato affermare che, in generale, i tempi di esecuzione delle due fasi citate siano più ridotti in ODI rispetto al PL/SQL.

5.1.3 Personalizzazione di implementazione degli estrattori dei dati

- **Descrizione**

  Questo indicatore valuta la possibilità di personalizzazione delle operazioni che vanno effettuate per implementare i flussi ETL. Viene comparata la facilità di realizzazione di estrattori "su misura" utilizzando i due strumenti considerati per la loro implementazione.

- **Confronto**

  In ODI, gli step che vengono effettuati durante l'esecuzione degli estrattori dipendono dal tipo di knowledge module che viene utilizzato per il lancio. Poichè è prevista la possibilità che l'utente possa modificare il codice che fa parte di ogni operazione prevista dai km, è garantita una buona possibilità di personalizzazione del flusso. In generale, comunque, essendo il software costituito da una serie di componenti predefinite, non sarà possibile riuscire a estendere a proprio piacimento le funzionalità offerte da ognuno di esse. Nel caso della programmazione

in PL/SQL, invece, potendo controllare direttamente la scrittura del codice, il programmatore potrà essere in grado di inserire, nei limiti delle possibilità del linguaggio, dei miglioramenti alla struttura del listato rispetto a come viene staticamente realizzata da ODI in base al km scelto dall'utente.

5.1.4 Impegno e profilo delle risorse necessarie alla realizzazione degli estrattori

- **Descrizione**

  L'indicatore preso in considerazione in questo caso propone un raffronto sul tipo di competenze tecniche che è necessario impiegare per poter effettuare il disegno e l'implementazione dei flussi ETL, utilizzando ODI o la programmazione in PL/SQL. Un ulteriore elemento utile di confronto riguarda l'impegno e la quantità di tempo necessaria, in relazione anche al numero di persone che occorre utilizzare, per raggiungere lo scopo, utilizzando le due soluzioni adottate.

- **Confronto**

  Come conseguenza diretta di quanto affermato in precedenza, si può sentenziare che le fasi di disegno e implementazione dei flussi ETL con ODI non richiedano, a valle della fase di setup della connessione, competenze tecniche di dettaglio relativamente alle sorgenti dati oggetto delle integrazioni, o una maggiore specializzazione relativa al linguaggio SQL, tramite cui ODI, realizza effettivamente le operazioni indicate dall'utente attraverso l'utilizzo degli strumenti grafici che mette a disposizione. Anche l'impegno della risorsa utlizzata, e il tempo totale di impiego, può risultare, di conseguenza, ridotto, utilizzando il software piuttosto che la metodologia tradizionale.

  Le risorse da impiegare per il disegno e lo sviluppo dei flussi in codice PL/SQL, invece, devono possedere le capacità per poter definire ed effettuare operazioni che talvolta possono risultare particolarmente complesse. Ad esempio, nel caso in cui vadano letti i dati da una determinata sorgente informativa, spesso è necessario, per portare a termine questo

step, che le risorse posseggano competenze specifiche che permettano loro di comprendere non solo gli aspetti procedurali, ma anche quelli tecnologici/architetturali.

5.1.5 Facilità di modellazione delle regole di business

- **Descrizione**

  I requisiti richiesti dal cliente conducono alla definizione di una serie di restrizioni, tramite i quali limitare i dati da considerare per l'estrazione. L'indicatore preso in considerazione compara le modalità di modellazione delle business rules utilizzando ODI e la programmazione in PL/SQL.

- **Confronto**

  ODI permette l'inserimento di business rules in modo semplice, grazie agli strumenti visuali che mette a disposizione dell'utente. E' possibile imporre diversi tipi di restrizioni: i filtri, le condizioni e i riferimenti. I filtri possono essere definiti durante la fase di creazione dell'interfaccia, nella scheda relativa al diagramma, digitando l'implementazione della regola da soddisfare per uno o più campi. Le condizioni vengono invece inserite direttamente nella scheda del modello della tabella per cui vogliono essere selezionati i record da estrarre. La differenza è che, in questo caso, ODI permetterà di ottenere informazioni di data quality per il risultato ottenuto, come il numero di record errati, che non sono stati estratti perchè non soddisfavano una determinata condizione, il motivo dell'esclusione del record considerato (trascrizione della business rules non soddisfatta), ecc. Infine i riferimenti permettono di definire dei vincoli di integrità referenziale tra colonne di due o più tabelle, riportando anche in questo caso statistiche sui record sottoposti a questa limitazione.

  Nel caso dello sviluppo dei programmi in PL/SQL, non si hanno a disposizione gli strumenti offerti per la gestione facilitata delle business rules, come visto nel caso precedente. L'implementazione dei requisiti

imposti dal cliente sui tipi di record da estrarre viene perciò realizzata unicamente contando sulle competenze del programmatore.

5.1.6 Gestione dei metadati

- **Descrizione**

  Di fondamentale importanza per il controllo dell'implementazione del processo ETL è la gestione dei metadati, che consente di mantenere dei riferimenti utili sugli object che vengono creati durante lo sviluppo degli estrattori. L'indicatore considerato permette il confronto tra le modalità di mantenimento dei metadati nel caso in cui i flussi vengano sviluppati tramite ODI o scrivendo codice in PL/SQL.

- **Confronto**

  La gestione dei metadati è certamente una delle caratteristiche che porta a considerare l'utilizzo di un software di integrazione dati come Oracle Data Integrator per l'implementazione dei propri flussi ETL. I suoi moduli grafici, in particolare il Designer, infatti, permettono di memorizzare quante più informazioni possibili riguardo gli oggetti creati all'interno dei progetti realizzati. Ai dati restituiti in automatico da ODI, per esempio relativi alla data di creazione o di modifica o al nome dell'utente che ha creato l'elemento preso in considerazione, si aggiungono quelli che lo stesso utente può immettere in fase di definizione dell'oggetto, ad esempio per chiarirne la motivazione di utilizzo. Oracle Data integrator permette inoltre la memorizzazione di altre informazioni sulle strutture dati o riguardo eventuali problemi rilevati durante le operazioni effettuate, ecc.

  In PL/SQL, i metadati deveono essere modellati dal programmatore in fase di scrittura del codice. Ciò che viene fatto, nella maggior parte dei casi, è inserire le informazioni richieste in dei file di log, che vengono costruiti parallelamente all'implementazione degli estrattori dei dati.

5.1.7 Gestione della documentazione e possibilità di generazione automatica delle informazioni

- **Descrizione**

  Questo parametro di confronto si riferisce alla possibilità di generazione di informazioni relative a disegno e implementazione degli estrattori. Nel caso in cui lo strumento utilizzato non possa essere in grado di generare automaticamente le informazioni richieste, si valuta la facilità di implementazione di documenti informativi da parte dello sviluppatore stesso.

- **Confronto**

  Nel momento in cui vengono completate le operazioni di disegno e sviluppo dei flussi ETL tramite l'utilizzo di ODI, lo sviluppatore ha immediatamente a disposizione importanti informazioni, utili in particolare per analizzare in maniera più semplice e veloce eventuali modifiche correttive o evolutive del software. Ad esempio, una parte importante di documentazione cui si fa riferimento è relativa al diagramma di connessione delle tabelle sorgenti o alla mappatura dei dati, che in ODI sono immediatamente consultabili aprendo la scheda relativa all'interfaccia che realizza l'estrazione dei dati.

  Il PL/SQL non genera automaticamente della documentazione, ciò non toglie che queste esigenze informative possano essere gestite direttamente dal programmatore, anche tramite l'utilizzo di tool specifici.

5.1.8 Supporto alle attività di debugging e testing

- **Descrizione**

  Un altro parametro che consente di valutare le prestazioni delle due soluzioni utilizzate è la stima della facilità di esecuzione delle attività di debugging e testing. Ciò può risultare direttamente correlato alla valutazione della complessità di individuazione della porzione di codice affetta da errori, rilevati a seguito dell'esecuzione del programma; questo è conseguenza, a sua volta, della quantità e qualità delle informazioni restituite dal compilatore o dal software in fase di debugging.

- **Confronto**

  In ODI l'esecuzione delle fasi di debugging e testing è facilitata dalla presenza di un modulo grafico specifico, chiamato Operator, analizzato più volte nei precedenti capitoli. Esso, una volta lanciata l'interfaccia, scompone il flusso da eseguire in più parti, permettendo di verificare l'esecuzione di ogni singola operazione definita dal knowledge module che si sta utilizzando. Per ognuna di esse, l'Operator riporta esito positivo o negativo, a seconda se sia andata o no a buon fine, visualizzando inoltre, in una scheda dedicata, la porzione di codice che è stata lanciata dall'agente per la sua esecuzione. Se per lo step in questione si è rilevato un errore, viene restituito un messaggio che riporta il motivo che ha causato l'interruzione del flusso. Seppur tutto ciò possa permettere, in generale, un'identificazione piuttosto rapida del problema che si è verificato, talvolta può accadere che venga restituita la scritta "comando SQL terminato erroneamente", che non permette di localizzare in tempi brevi quello che, spesso, potrebbe anche essere un banale errore di digitazione. In generale, quindi, ODI sfrutta l'Operator per riportare all'utente i codici e messaggi di errore restituiti dal compilatore SQL, ma non aggiunge altre indicazioni utili per la localizzazione dei bug. Per quanto riguarda la fase di testing, gli strumenti grafici messi a disposizione da ODI, permettono di verificare facilmente la consistenza delle informazioni migrate nel supporto di destinazione, permettendo di individuare eventuali bachi che portino a un malfunzionamento del flusso e delle informazioni trasferite nei supporti di destinazione.

  Per quanto riguarda il PL/SQL, invece, proprio il fatto di essere liberi dai rigidi moduli grafici di ODI, consente di personalizzare le fasi di debugging e testing a seconda delle proprie esigenze; per questo, se il programmatore è particolarmente esperto ed ha il supporto delle funzionalità di debugging dell'ambiente di sviluppo, i tempi della loro esecuzione possono, in generale, essere più brevi rispetto al caso di utilizzo del software. Il vantaggio di poter operare direttamente sulle linee di codice, infatti, permette di dividere a piacimento l'esecuzione del flusso, inserendo in ogni step variabili di controllo che permettano di individua-

re rapidamente il problema che è occorso in fase di debugging, o quello che ha causato l'inserimento di informazioni errate, individuate durante la fase di testing. Si può concludere, dunque, che la quantità di impegno dedicata a queste attività, nel caso della programmazione tradizionale, sia in generale fortemente correlata alle capacità e competenze dello sviluppatore.

5.1.9 Gestione del rilascio in produzione dei flussi ETL

- **Descrizione**

  Una volta che i flussi ETL vengono implementati, deve essere prevista la possibilità che essi vengano eseguiti in un ambiente di produzione, dopo aver effettuato tutte le attività di testing necessarie. L'indicatore in questione compara la gestione del rilascio degli estrattori implementati per le due soluzioni prese in considerazione.

- **Confronto**

  Package e scenari sono gli strumenti offerti da ODI per la gestione del rilascio in produzione dei flussi ETL implementati. Come visto in precedenza, i package consentono di eseguire più interfacce con un unico lancio, permettendo inoltre di avviare in contemporanea anche altri oggetti, come delle procedure, e di definire l'ordine di avvio di ognuno degli elementi indicati. Lo scenario permette l'esecuzione della singola interfaccia o di un package fuori dall'ambiente ODI, affidandone l'avvio al sistema operativo piuttosto che all'agente utilizzato dal software. L'utilizzo degli strumenti grafici messi a disposizione dal software facilita l'esecuzione di questa fase e permette la minimizzazione degli errori che possono occorrere durante essa.

  Il rilascio in produzione nel caso del PL/SQL comporta l'accorpamento manuale in package dei programmi e altri script necessari per l'implementazione dei flussi ETL, di modo che essi possano essere lanciati in esecuzione contemporanenamente. In questo caso tutte le operazioni e i controlli legati alla gestione del pacchetto che si deve rilasciare devono essere interamente effettuate dal programmatore.

5.1.10 Considerazioni finali

A completamento del paragrafo viene visualizzato un grafico a radar che riassume le prestazioni offerte da Oracle Data Integrator e la programmazione in PL/SQL in merito ai parametri di paragone considerati in precedenza. Ai due strumenti, tra cui si effettua la comparazione, è stata assegnata una valutazione (1=non sufficiente, 2=sufficiente, 3=buono), a seguito di quanto evidenziato nei paragrafi che hanno preceduto queste considerazioni finali.

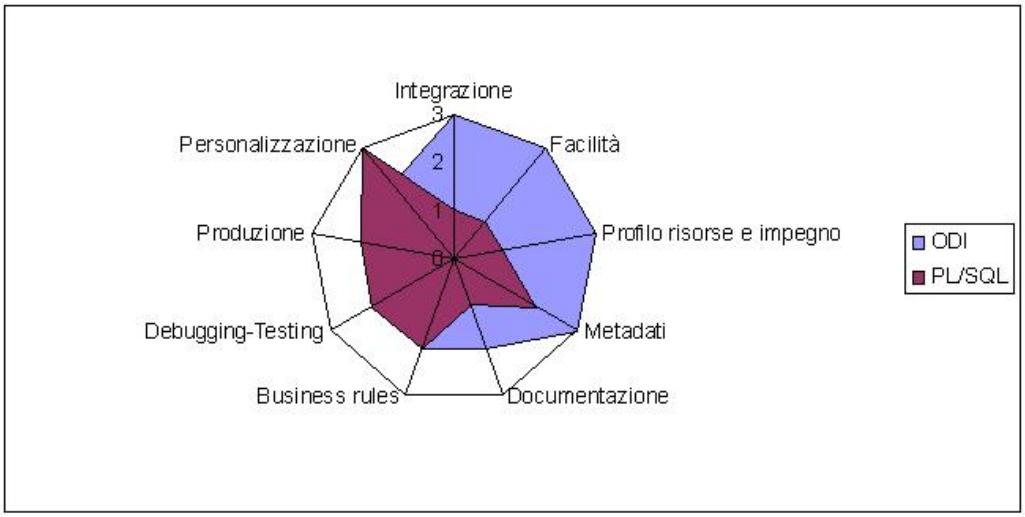


Figura 5.1: Disegno ed implementazione dei flussi ETL

5.2 Fase di runtime (Production)

La seconda fase che costituisce un progetto di implementazione di flussi ETL consiste nell'esecuzione periodica delle interfacce o dei programmi implementati, in maniera da alimentare ad intervalli predefiniti il data warehouse. Nelle sottosezioni che fanno parte di questo secondo paragrafo si procede all'analisi dei parametri da prendere in considerazione per questo step, cui succede un confronto dettagliato, rispetto al punto di vista preso in considerazione, tra quanto fatto con Oracle Data Integrator e con lo sviluppo dei programmi in PL/SQL.

5.2.1 Efficienza di implementazione degli estrattori dei dati

- **Descrizione**

  Il primo importante termine di paragone da considerare per la fase di runtime consiste nella valutazione dell'efficienza dell'esecuzione periodica dei flussi implementati, in termini di prestazioni ottenute relativamente alla velocità di esecuzione degli stessi o alla quantità di memoria impiegata.

- **Confronto**

  ODI consente di ottenere buone prestazioni per quel che riguarda l'esecuzione dei flussi ETL. Tuttavia, in particolar modo se la tecnologia delle sorgenti informative che ospiteranno le tabelle di destinazione è Oracle, l'esecuzione di programmi scritti nel suo linguaggio nativo, il PL/SQL, risulterà, in generale, più performante dal punto di vista considerato.

  Rispetto agli altri tool presenti sul mercato, potrebbe essere citato come vantaggio del PL/SQL anche il risparmio sul trasferimento dei dati verso il server in cui il software ETL effettua le trasformazioni e poi di nuovo verso il database di destinazione. Ciò non avviene però per ODI, per il fatto che esso implementa un'architettura di tipo E-LT diversa da quella utilizzata dagli altri tool (si rimanda alla parte dedicata ad Oracle Data Integrator, nel primo capitolo, per maggiori dettagli su questo argomento).

5.2.2 Gestione della consistenza dei dati estratti

- **Descrizione**

  Questo parametro di valutazione si riferisce alla possibilità offerta dai due strumenti considerati di controllo/tracciatura dei record che sono stati interessati dal processo di estrazione effettuato. L'obiettivo è quello di valutare la capacità di assicurare che non vengano caricate informazioni erronee o duplicate nel data warehouse e quindi digarantire la consistenza dei dati migrati verso il sistema di destinazione.

- **Confronto**

Figura 5.2: Inserimento condizione

Oracle Data Integrator offre degli strumenti che consentono di individuare i record che violano determinati vincoli imposti dall'utente, mantenendone traccia di modo che, eventualmente si possa inserirli in dei report riepilogativi. Supponiamo, ad esempio, che si vogliano effettuare dei controlli di integrità dei dati per verificare quali record vengano filtrati rispetto a un determinato vincolo imposto, e quali altri, invece, vengono trasferiti nella tabella di destinazione. In un esempio pratico, si ipotizza di voler tenere traccia soltanto degli utenti la cui età sia maggiore dei 21 anni. La figura 5.2 mostra l'inserimento della condizione proposta.

Una volta lanciata un'interfaccia che ha fra le tabelle sorgenti quella per cui è stato imposto il vincolo, sarà possibile verificare la quantità di record che non sono stati migrati e i valori di tutti i campi dei record in questione (figura5.3).

Nell'immagine visualizzata si nota che nell'archivio che ospita i dati che non sono stati trasferiti nei supporti di destinazione vengono mantenuti in totale nove record. Di questi, otto sono considerati errati poichè il valore relativo al loro campo dell'età è inferiore al minimo richiesto, mentre la prima riga si riferisce a un record che non ha superato un controllo di integrità referenziale.

Nel caso del PL/SQL, il programmatore può scegliere di implementare

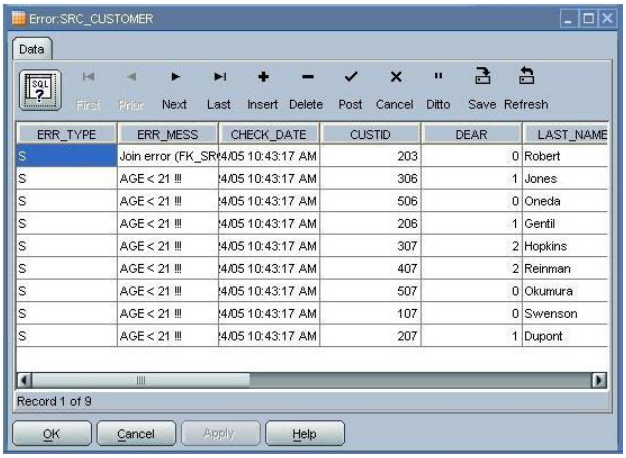


Figura 5.3: Visualizzazione record errati

le funzionalità viste per il caso di ODI manualmente ed estendendo il codice scritto per l'implementazione dei flussi. Una possibilità potrebbe consistere nella realizzazione di nuove tabelle per il controllo degli errori, in cui migrare i record che vengono filtrati da determinate condizoni imposte, come viene fatto in automatico dal software. Anche in questo caso, quindi, il maggior tempo richiesto per l'implementazione di operazioni del genere, può essere parzialmente controbilanciata dalla possibilità di maggiore personalizzazione delle stesse.

5.2.3 Supporto alle operazioni di data quality

- **Descrizione**

  La qualità dei dati è un fattore critico per il successo delle iniziative di business intelligence aziendali. I dati di scarsa qualità possono facilmente e rapidamente propagarsi all'interno degli altri sistemi. Se le informazioni condivise dalle molteplici istanze applicative aziendali risultano essere contraddittorie, inconsistenti o imprecise allora anche le relazioni con clienti, fornitori e partner si baseranno su informazioni inesatte ed approssimative, innalzando i costi e riducendo la credibilità aziendale. Questo indicatore valuta le possibilità offerte dai due strumenti considerati per il supporto alla data quality, in cui rientrano

anche le operazioni di controllo della consistenza dei dati analizzate in precedenza.

- **Confronto**

  Insieme ad ODI viene installato un software chiamato "Oracle Data Profiling e Quality", che permette all'utente di eseguire indagini approfondite di data quality e di ottenere diversi tipi di statistiche riguardo i record che sono stati trasferiti in seguito all'esecuzione delle interfacce implementate (data profiling). Sia per i record estratti che per quelli scartati e mantenuti in un archivio a parte, è possibile ottenere informazioni come la lunghezza degli attributi, valori massimi e minimi dei campi, percentuale di distribuzione dei loro valori dei record migrati, ecc. Lo strumento permette, inoltre, di realizzare report, grafici o altri tipi di oggetti riguardo le statistiche estratte. A partire da queste informazioni, è possibile individuare facilmente eventuali errori presenti anche nelle sorgenti informative da cui vengono estratti i dati prima della fase di trasformazione, ovvero di effettuare operazioni di pulizia prima ancora di implementare il processo di ETL. Ad esempio il software sarà in grado di isolare quei record per cui si è avuto un errore di digitazione in uno dei suoi campi, dopo aver paragonato il numero di occorrenze dei diversi tipi di valori con cui un determinato campo viene valorizzato. Oracle Data Quality e Profiling è stato citato soltanto in questa occasione poichè, pur essendo un componente importante della suite di data integration di Oracle, il suo utilizzo non è previsto come parte integrante del presente lavoro di tesi.

  Per quanto riguarda il PL/SQL, valgono le stesse valutazioni fatte per il parametro di valutazione precedente; ovvero tali funzionalità devono essere implementate dal programmatore estendendo il codice scritto per l'implementazione dei flussi.

5.2.4 Gestione della sicurezza

- **Descrizione**

Quando si pianifica un sistema di sicurezza per un progetto di integrazione di dati, si cerca di assicurare la tutela di due tipi di dati: i dati di sviluppo, controllando gli accessi effettuati ai development object, e i dati di produzione, controllando gli accessi al sistema operazionale sorgente e a quello di destinazione. Questo parametro di valutazione analizza la gestione della sicurezza per entrambi questi tipi di dati, paragonando le prestazioni fornite dal software e dai programmi scritti in PL/SQL.

- **Confronto**

  Oracle Data Integrator mette a disposizione un modulo grafico separato, chiamato Security Manager, per l'implementazione delle politiche di sicurezza. Tramite l'utilizzo di questo strumento, è possibile innanzitutto definire password di accesso per l'utilizzo del tool e di alcune funzionalità all'interno di esso, nonchè definire utenti abilitati all'accesso e profili per definire cosa possono e non possono fare, in termini di creazione, semplice lettura o modifica dei vari object che esistono all'interno dei progetti realizzati. Particolare importanza, a questo proposito, assume l'utilizzo dei contesti, presentati nei capitoli precedenti della tesi. Come si è visto, essi realizzano l'astrazione che consente agli sviluppatori di lavorare in un ambiente logico e poi eseguire il loro lavoro negli ambienti fisici cui essi hanno accesso con i loro privilegi. Ciò permette, ad esempio, ad un utente di avere accesso agli object sviluppati, ma non all'ambiente di produzione o di testing.

  Nel caso della realizzazione dei programmi in PL/SQL che, una volta lanciati, eseguono le operazioni di caricamento dei cubi, le politiche di sicurezza vengono implementate scrivendo del codice supplementare, che permetta di definire gli utenti autorizzati per l'esecuzione dei flussi e l'estrazione dei dati dalle sorgenti informative in questione.

5.2.5 Considerazioni finali

A completamento del paragrafo viene visualizzato un grafico a radar che riassume le prestazioni offerte da Oracle Data Integrator e la programmazione in PL/SQL in merito ai parametri di paragone considerati in precedenza. Ai

due strumenti, tra cui si effettua la comparazione, è stata assegnata una valutazione (1=non sufficiente, 2=sufficiente, 3=buono), a seguito di quanto evidenziato nei paragrafi che hanno preceduto queste considerazioni finali.

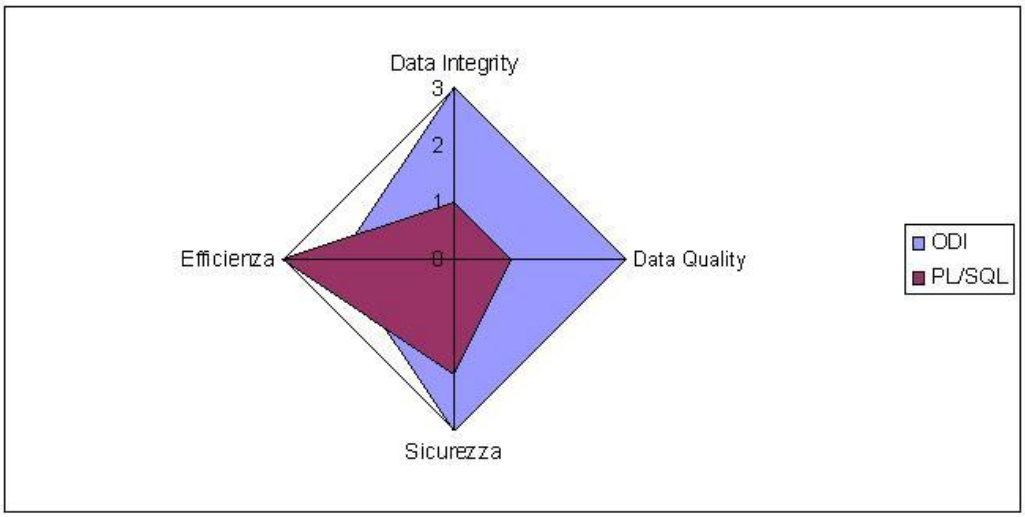


Figura 5.4: Fase di runtime

5.3 Manutenzione correttiva ed evolutiva dei flussi ETL

La terza e ultima fase in cui può essere metodologicamente scomposto un processo di ETL è quella della manutenzione dei flussi che sono stati in precedenza implementati, conseguenza, evidentemente, di un cambio dei requisiti per i dati da estrarre, o di una modifica dettata dalla localizzazione di errori a seguito dell'esecuzione dei programmi o delle interfacce. In caso di cambiamenti al flusso, devono essere rieffettuati tutti i passi che fanno parte del ciclo di vita del software, dalla nuova analisi ai test che permettono di verificafe se le nuove modifiche hanno realmente avuto gli effetti desiderati.

Di seguito vengono presentati gli indicatori che possono essere presi in considerazione in riferimento alle operazioni da effettuare citate, per eseguire

il confronto tra ODI e la programmazione in PL/SQL anche da questo punto di vista.

5.3.1 Facilità della misurazione dell'entità di impatto dovuto al cambiamento richiesto

- **Descrizione**

  L'indicatore in questione prende in considerazione, in caso di modifica ai requisiti da parte del cliente riguardo il tipo di dati da estrarre o di localizzazione di un bug, il tempo necessario per l'individuazione del cambiamento al disegno e all'implementazione dei flussi ETL che consenta di soddisfare le nuove esigenze o di correggere eventuali errori.

- **Confronto**

  In generale, l'implementazione grafica offerta da ODI consente di individuare le modifiche da effettuare al flusso ETL già sviluppato, in maniera, il più delle volte, immediata e di misurare facilmente l'entità dell'impatto delle stesse. Ad esempio, nel caso dell'aggiunta della dimensione relativa alla città del fornitore, per il cubo degli ordini dei materiali, il fatto che nel diagramma dell'interfaccia relativa a quel processo di ETL comparissero tutte le tabelle sorgenti interessate e, per ognuna di esse, l'elenco dei campi presenti, ha permesso di stabilire immediatamente la facilità della soddisfazione dei nuovi requisiti. Può essere quindi, in questo caso, sfruttata parte della documentazione sul progetto generata automaticamente dal software.

  Immediato raffronto non si può avere invece nel caso dei concurrent sviluppati in PL/SQL. In questo caso, infatti, non è possibile sfruttare le potenzialità grafiche offerte dal software per misurare l'entità delle modifiche; ciò dovrà quindi essere fatto in maniera tradizionale dal progettista, che in questo caso potrebbe peraltro essere una persona diversa dallo sviluppatore.

5.3.2 Comparazione del tempo necessario alla revisione degli estrattori

- **Descrizione**

  Una volta che sono state individuate le modifiche al disegno dei flussi che permettano di accogliere le nuove richieste del cliente, il passo successivo consiste nello sviluppo tecnico dei nuovi requisiti da rendere operativi o degli errori da correggere. Un parametro di valutazione importante è, quindi, quello che esamina la complessità di esecuzione di questo step e che permette il paragone le soluzioni offerte, a questo proposito, da ODI e dai programmi generati in PL/SQL.

- **Confronto**

  Le stesse cose dette per il parametro di valutazione precedente, possono essere ripetute anche in questo caso. Infatti è grazie ai suoi moduli grafici che ODI riesce a fornire ottime prestazioni anche in riferimento al tempo dedicato per l'implementazione delle nuove modifiche. Riprendendo l'esempio della modifica del cubo degli ordini, infatti, osservando il diagramma dell'interfaccia relativa, ci si è accorti che per implementare le nuove modifiche era sufficiente mappare il campo richiesto con il valore contenuto nella colonna di una delle tabelle che veniva letta, il che avviene tramite l'utilizzo di un'interfaccia grafica intuitiva.

  Nel caso del PL/SQL, data la distanza tra le fasi di design e implementazione, nel caso di modifiche ai requisiti, il documento contenente i cambiamenti verrebbe passato dal progettista allo sviluppatore, che provvederebbe poi a implementarle scrivendo le righe di codice supplementari. Ciò comporta la necessità di maggiore impegno e quantità di tempo richiesta alle risorse impiegate.

5.3.3 Strumenti a disposizione per l'integrazione del ciclo di vita del software

- **Descrizione**

  A partire dalle considerazioni che emergono dalla comparazione degli strumenti utilizzati in base ai due parametri di valutazione visti in pre-

cedenza, il presente indicatore considera le prestazioni offerte a proposito dell'integrazione dell'intero ciclo di vita del software, in seguito al suo rilascio in produzione, nel momento in cui emerge l'esigenza di effettuare nuove modifiche ai flussi implementati. Nello specifico, si indaga sulla necessità di reiterare le stesse operazioni nelle diverse fasi del processo che devono essere eseguite per l'integrazione delle nuove modifiche, dal nuovo disegno alla fase di rilascio in produzione della nuova implementazione dei flussi, e al tempo richiesto per il completamento di tutte queste operazioni.

- **Confronto**

  Quanto detto in precedenza porta come risultato, per quel che riguarda ODI, a un accorciamento del tempo dedicato all'integrazione delle modifiche ai requisiti o alla correzione dei bug individuati, in particolare per quanto riguarda il gap tra la fase di progettazione e quella di sviluppo, che potrebbe accorciarsi, fino al punto che le stesse possano essere effettuate da un'unica persona. In questo caso, infatti, lo stesso strumento potrebbe essere utilizzato per capire, per implementare e per documentare.

  Nel caso della programmazione in PL/SQL, invece, in generale, le fasi di progettazione e sviluppo sembrano essere più distanti tra loro. Questo è dovuto principalmente al fatto che le competenze che servono per la scrittura del codice sono molto specifiche, di conseguenza queste due fasi saranno prabilmente effettuate da due risorse che possiedono due diversi tipi di conoscenze: una più funzionale e l'altra più tecnologica. L'integrazione delle modifiche ai flussi ETl richiederà, quindi, in generale, tempi più lunghi rispetto al caso precedente.

5.3.4 Considerazioni finali

A completamento del paragrafo viene visualizzato un grafico a radar che riassume le prestazioni offerte da Oracle Data Integrator e la programmazione in PL/SQL in merito ai parametri di paragone considerati in precedenza. Ai due strumenti, tra cui si effettua la comparazione, è stata assegnata una

valutazione (1=non sufficiente, 2=sufficiente, 3=buono), a seguito di quanto evidenziato nei paragrafi che hanno preceduto queste considerazioni finali.

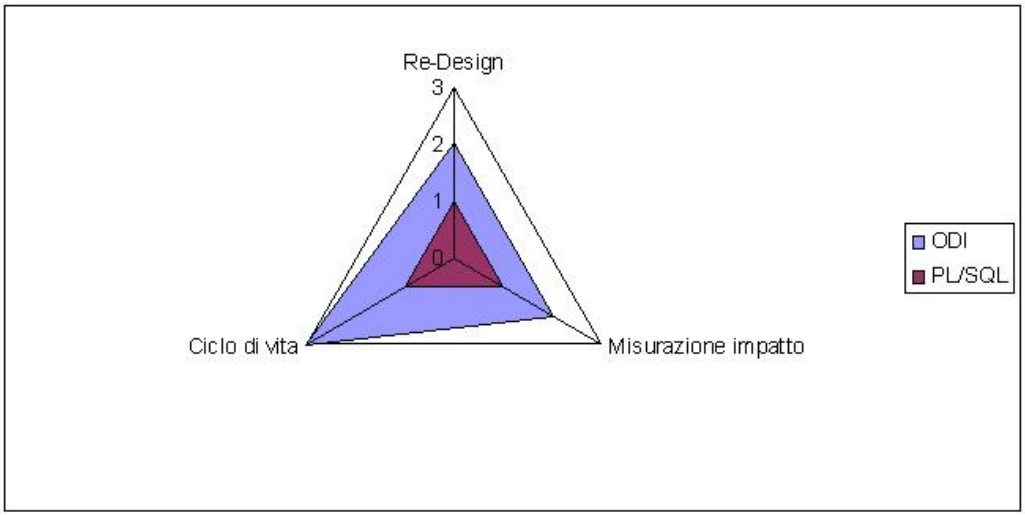


Figura 5.5: Manutenzione dei flussi ETL

Capitolo 6

BENCHMARK: PARAGONE ECONOMICO

Scopo di questo capitolo è effettuare una valutazione delle due soluzioni alternative, non più prendendo in considerazione gli aspetti tecnici ma, bensì, quelli di tipo economico che sono il riflesso diretto dell'impegno di risorse necessario all'implementazione degli estrattori ETL. L'analisi sarà utile per valutare, se possibile, in quali ambiti sia preferibile una delle due soluzioni rispetto all'altra. Per esprimere tali considerazioni, si è scelto di effettuare una scomposizione metodologica delle attività necessarie alla realizzazione degli estrattori e, analogamente a quanto fatto nella valutazione degli aspetti operativi, di dare un peso alle singole attività, in questo caso tramite una loro valutazione di impegno espresso in giorni-uomo. Tale stima è stata effettuata mediante interviste con il program-manager del progetto. A valle della valutazione dell'impegno necessario al completamento del lavoro, è stata elaborata una valorizzazione media relativa al costo necessario all'implementazione di un estrattore nei due scenari con il fine ultimo sopra enunciato, ovvero poter esprimere un giudizio di "preferibilità" di una delle due modalità rispetto ad un contesto teorico.

6.1 Introduzione

Nel capitolo introduttivo della tesi si è ipotizzata la possibilità di utilizzare l'indice del ROI (return on investment) quale criterio di comparazione dei due scenari, utile anche per l'elaborazione di un criterio decisionale a supporto della scelta "Make or Buy". In pratica, considerate le caratteristiche del progetto che è stato selezionato come caso di studio, in cui l'implementazione degli estrattori costituisce una attività collaterale, non è stato ritenuto significativo impostare un'analisi quantitativa utilizzando l'indicatore citato. Si è però ritenuto applicabile effettuare l'analisi di comparazione dei costi medi di sviluppo relativi ai due scenari scenari e ciò ha condotto ad elaborare un criterio di "preferenza" in funzione di un contesto teorico, esprimibile tramite una break even analisys. Ciò consente di raggiungere ugualmente lo scopo che ci si prefiggeva per il tipo di analisi inizialmente prevista e cioè di valutare in quali casi, a seconda della complessità del data warehouse da alimentare nonché del numero di cubi da realizzare, sia preferibile una soluzione make rispetto ad una buy.

6.2 Scomposizione del progetto in sotto-attività

In questo paragrafo vengono individuate le fasi in cui può essere idealmente metodologicamente scomposta l'attività di implementazione di un flusso ETL. Sia nel caso di realizzazione degli estrattori tramite utilizzo di ODI che attraverso la scrittura dei programmi in PL/SQL, vengono individuate le seguenti attività:

- **Analisi dei requisiti e scrittura della documentazione tecnica**

  Questa attività comprende tutte le operazioni che precedono la parte di implementazione tecnica degli estrattori. Il colloquio con l'utente permette di definire il requisito (nel caso specifico del cubo, qual'è il fatto o processo che deve essere analizzato e da quali punti di vista, o dimensioni, bisogna effettuare l'indagine). In seguito, a partire dal documento dei requisiti, vengono individuate le sorgenti informative da cui reperire i dati necessari e si procede ad effettuare la progettazione concettuale e

logica del cubo. Tutte le operazioni citate devono fornire come output la documentazione che costituirà il punto di partenza della successiva fase di sviluppo degli estrattori.

- **Implementazione**

  Questa fase è costituita dalle operazioni di estrazione dei dati dalle sorgenti informative, trasformazione nella staging area e caricamento nelle tabelle di destinazione, tenendo conto delle specifiche definite nella documentazione. Questa parte comprende inoltre lo sviluppo degli script di test necessari per la verifica delle informazioni trasferite. Nei capitoli 3 e 4 è stato effettuato lo sviluppo dei due cubi presi in considerazione prima utilizzando i moduli grafici offerti da Oracle Data Integrator e successivamente attraverso la realizzazione di programmi in PL/SQL.

- **Test di integrazione**

  In seguito all'implementazione tecnica si procede alla fase di testing, il cui scopo principale è la verifica delle informazioni che vengono caricate nell'archivio target. Questa attività si conclude nel momento in cui si è certi che i dati caricati siano coerenti rispetto a quelle che erano le richieste iniziali del cliente.

- **User acceptance test**

  Tale attività consiste nella ripetizione dei test con la partecipazione degli utilizzatori finali dei cubi. Tali collaudi formali permettono la verifica con il cliente finale della corrispondenza del prodotto al requisito e dell'effettivo funzionamento degli estrattori.

- **Rilascio in produzione**

  Una volta terminata la fase di testing, gli estrattori vengono infine raggruppati in package o scenari e rilasciati in produzione. Le interfacce o i programmi verranno periodicamente lanciati in maniera tale da garantire l'aggiornamento delle informazioni contenute nel data warehouse.

6.3 Modello di valutazione necessità di impegno

Obiettivo di questo paragrafo è la modellizzazione dei due scenari oggetto della valutazione. Essa è mirata alla valutazione della necessità di "lavoro" necessario alle attività di sviluppo in ognuno dei due scenari.

In figura 6.1 si mostra lo scenario relativo alla programmazione tradizionale.

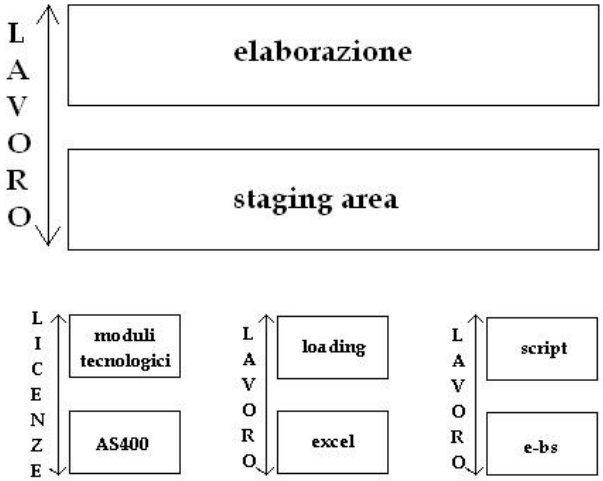


Figura 6.1: Scenario PL/SQL

In questo caso, che rappresenta l'ipotesi "Make", si evidenzia un maggior numero di componenti, principalmente legate a problematiche di integrazione tra sistemi, che devono essere sviluppate direttamente dal programmatore. Egli deve essere in grado di gestire, sviluppando il codice necessario, tutte le fasi che compongono l'implementazione del flusso ETL, dalla lettura dei dati dalle sorgenti, alle trasformazioni effettuate nella staging area, al caricamento dei dati trasformati nelle tabelle di destinazione.

In figura 6.2 viene mostrato il dettaglio relativo allo sviluppo attraverso l'utilizzo del software.

In questo caso è evidente come lo strumento software renda disponibili i connettori verso le fonti dati riducendo il lavoro di sviluppo necessario alla realizzazione dei driver specifici. I moduli grafici offerti da ODI semplificano la parte implementativa e consentono di conseguenza una riduzione dei tempi

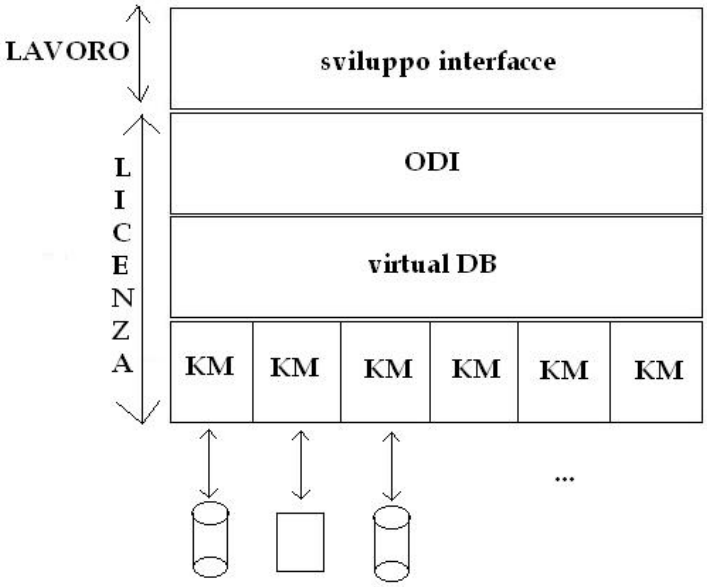


Figura 6.2: Scenario ODI

dedicati a questa importante fase del ciclo di vita del software.

6.4 Valutazione costi di sviluppo e di acquisizione delle licenze

Dopo aver scomposto in fasi l'intero processo di sviluppo del flusso ETL per il caricamento di un cubo nel data warehouse, si procede in questo paragrafo alla stima del lavoro necessario, espresso in giorni-uomo, per il completamento di tutte le operazioni preventivate (per l'implementazione di un cubo di media complessità), sia nel caso di sviluppo di programmi in PL/SQL che nel caso di utilizzo del software Oracle Data Integrator. Le stime elaborate saranno ovviamente figlie delle considerazioni effettuate nel capitolo precedente in merito alle differenze tecnologiche esistenti tra le due soluzioni adottate. Le informazioni numeriche elaborate saranno utili nel paragrafo successivo al fine di determinare il criterio di valutazione del contesto make or buy affinchè sia possibile la comparazione dal punto di vista economico (ciò corrisponde ad individuare il break even point).

- **Analisi dei requisiti e scrittura della documentazione tecnica**

  - **PL/SQL**: 3-5 gg. per l'analisi dei requisiti e 2 gg. per la documentazione
  - **ODI**: 3-5 gg. per l'analisi dei requisiti e 1 gg. per la documentazione

- **Implementazione**

  - **PL/SQL**: 5 gg. per l'implementazione e 1 gg. per gli script di test
  - **ODI**: 2.5 gg. per l'implementazione e 1 gg. per gli script di test

  In questo caso la stima delle giornate lavorative necessarie è fortemente correlata al numero e alle tipologie delle sorgenti di dati a cui è necessario connettersi. Come sottolineato nel capitolo dedicato al paragone tecnologico, in caso di tabelle sorgenti non accessibili direttamente attraverso una connessione trasparente del database, la scrittura dei driver necessari per il collegamento aumenterebbe considerevolmente il divario tra le due soluzioni studiate relativamente alla quantità di tempo da dedicare al problema.

- **Test di integrazione**

  - **PL/SQL**: 2 gg.
  - **ODI**: 1 gg.

- **User acceptance test**

  - **PL/SQL**: 0.5 gg.
  - **ODI**: 0.5 gg.

- **Rilascio in produzione**

  - **PL/SQL**: 0.5 gg.
  - **ODI**: 0.5 gg.

Nell'ambito del progetto di sviluppo, il lavoro utile è quello direttamente finalizzato alla sua realizzazione. E' usuale prendere in considerazione fattori di "contingency" e/o "overhead", ovvero la stima dedicata alla gestione di

situazioni e/o problemi non preventivabili a priori (es. bug fixing, non disponibilità delle risorse in base alla necessità, attività di coordinamento). Nel caso dell'implementazione tramite PL/SQL, l'overhead si può stimare nell'ordine del 20 % delle ore preventivate, mentre per ODI si prevede essere intorno al 10%. La principale ragione della differenza è, nel caso PL/SQL, la necessità di coinvolgimento di un team di lavoro più esteso (analisti funzionali e programmatori).

Il numero di giornate lavorative individuate per la realizzazione degli estrattori viene tradotto nel corrispondente costo utilizzando la tariffa giornaliera impiegata come media di riferimento (1 gg. = 550 euro). Tali costi, sia in un caso che nell'altro, sono dipendenti dal numero di flussi ETL che devono essere realizzati (quelli determinati nel precedente paragrafo si riferiscono, come già sottolineato, alla realizzazione di un solo estrattore di media complessità), e pertanto vengono definiti costi variabili. La determinazione dei costi fissi, non direttamente correlati al numero di interfacce da sviluppare, comporta nel caso di ODI la considerazione della spesa sostenuta per l'acquisizione della licenza del software, mentre invece l'approccio della programmazione tradizionale non genera alcun esborso monetario indipendente dalla quantità di lavoro effettuata.

Il prezzo della licenza di ODI (in base al listino dei prodotti Oracle attivo al momento della redazione della presente tesi) è di 23 mila euro con metrica per processore. Come architettura di riferimento, in base alla quale nel capitolo successivo verrà determinato il costo del software, si seleziona quella di riferimento del progetto Erasmus, in cui Oracle Data Integrator è presente in due calcolatori, ognuno dotato di quattro processori.

6.5 Break even analysis

Abbiamo già evidenziato, nel paragrafo precedente, la diversa struttura dei costi per i due casi analizzati. In questa sede l'obiettivo è quello di individuare il cosiddetto punto di pareggio del progetto, ovverosia il volume di flussi ETL da realizzare in corrispondenza del quale sia conveniente l'impiego del prodotto che, come abbiamo visto precedentemente, riduce il lavoro necessario ma comporta un costo di licenza.

A partire dalle informazioni numeriche ottenute nei paragrafi precedenti, si procede di seguito alla visualizzazione dei passaggi che sono necessari per il calcolo del break even point.

$(5+2+5+1+2+0.5+0.5)\cdot(550)\cdot x = 184000 + (5+1+2.5+1+1+0.5+0.5)\cdot(550)\cdot x$

$8800 \cdot x = 184000 + 6325 \cdot x$

$\mathbf{x} \cong 74$

Dall'analisi si evince che implementando 74 flussi ETL utilizzare il software o la programmazione tradizionale comporta esattamente gli stessi costi. Se all'interno del progetto devono essere realizzati meno flussi ETL è conveniente sviluppare i programmi in PL/SQL, mentre per progetti più complessi risulta preferibile l'acquisizione della licenza di ODI.

La situazione può essere sintetizzata tramite la rappresentazione grafica mostrata in figura 6.3.

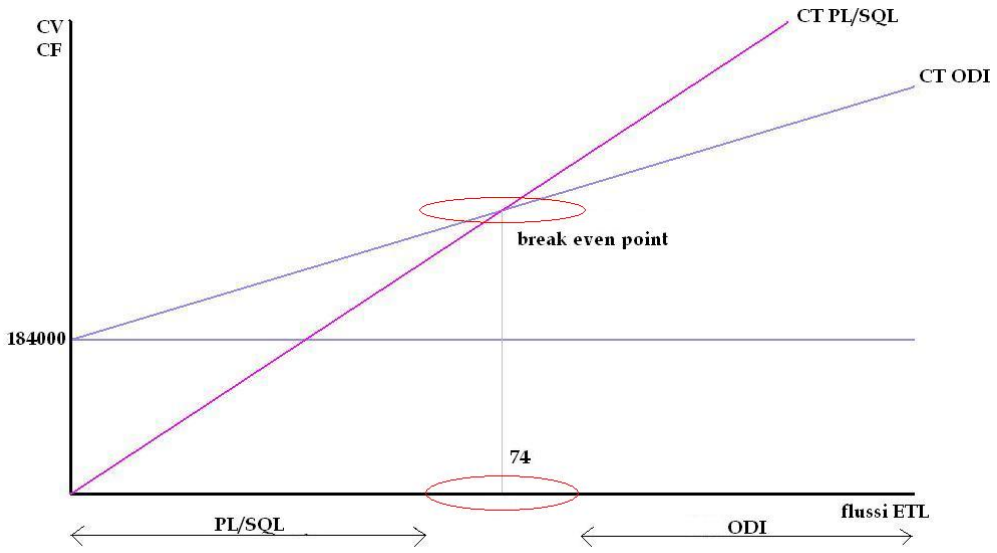


Figura 6.3: Break even analysis

Avendo fino ad ora basato lo studio su delle stime, e data in particolare la variabilità dell'analisi rispetto alla complessità delle sorgenti dati da collega-

re, si indica tramite l'area evidenziata dall'ellisse l'intervallo di interfacce da realizzare per cui ponderare con maggiore attenzione la scelta di implementare le stesse tramite programmi scritti in PL/SQL o utilizzando il software Oracle Data Integrator.

Conclusioni

L'obiettivo del presente lavoro di tesi è stato quello di effettuare l'analisi comparativa tra due approcci alternativi applicabili al processo di disegno e realizzazione degli "estrattori" per l'alimentazione di un DataWarehouse. L'esigenza che ha generato il lavoro è stata l'opportunità di identificare una metodologia ed una conseguente metrica per valutare se e dove l'adozione di un approccio sia preferibile rispetto all'altro. L'indagine è stata inoltre stimolata dalla attuale situazione di mercato che rende disponibili prodotti dedicati quale alternativa a soluzioni di implementazione classiche.

Al fine di basare lo studio anche su valutazioni empiriche, come parte integrante del lavoro sono stati implementati due estrattori preposti alla trasformazione ed al caricamento dei dati in un data warehouse (quello implementato nel progetto utilizzato come riferimento), prima scrivendo programmi in PL/SQL e poi utilizzando il software Oracle Data Integrator. A partire da queste attività si è potuto eseguire il confronto tra le due soluzioni adottate, comparando gli aspetti tecnici e quelli economici.

Le considerazioni tecniche, esposte nel quinto capitolo, hanno consentito di individuare gli elementi che propendono a favore di ciascuno dei due strumenti. L'analisi è stata divisa in tre parti, perchè tre sono le fasi in cui può essere metodologicamente scomposto il ciclo di vita del software da realizzare. Per ciascuna di esse sono stati identificati un certo numero di indicatori, o elementi di confronto, in base ai quali è stato possibile assegnare un giudizio sulla realizzazione dei flussi con PL/SQL e con Oracle Data Integrator. Per ciò che riguarda la prima fase, analisi di dettaglio e sviluppo, è emerso che il software offre dei vantaggi considerevoli rispetto alla programmazione tradizionale specialmente in termini di facilità di implementazione dei flussi

quando è necessario integrare più sorgenti di dati eterogenee; ciò ha come diretta conseguenza la possibilità di impiegare, nel caso di realizzazione degli estrattori col software, risorse meno esperte dal punto di vista della conoscenza del linguaggio e delle problematiche di connessione con i diversi tipi di sorgenti dati. Un vantaggio della programmazione PL/SQL è invece la possibilità di maggiore personalizzazione di implementazione degli estrattori. Durante l'analisi della seconda fase, run-time o production, si è rilevato in particolare che il software offre maggiori garanzie in termini di qualità ed integrità di dati, e consente quindi di avere un controllo maggiore sui record che vengono migrati e su quelli che invece, per qualche ragione, vengono filtrati e quindi non trasferiti nelle tabelle di destinazione. Il PL/SQL, d'altra parte, consente di ottenere un livello di efficienza maggiore di esecuzione dei flussi, in particolare se le tecnologie da connettere sono di tipo Oracle. Infine, per quel che riguarda la terza fase, manutenzione evolutiva e correttiva dei flussi, è emerso che, in caso di modifiche da effettuare ai flussi, a seguito di nuove richieste del cliente o di errori rilevati durante le esecuzioni degli estrattori, il software permette di rendere più efficaci i processi di re-ingegnerizzazione e re-implementazione del flusso, poichè il suo utilizzo, in sostanza, consente di ridurre considerevolmente il gap tra la fase di analisi e quella di sviluppo.

L'analisi economica ha avuto come scopo quello di definire una metrica per poter oggettivare la preferibilità di una delle due modalità rispetto ad un contesto teorico. L'analisi è stata condotta valutando il lavoro necessario (espresso in giorni/uomo), e quindi i costi corrispondenti, necessari al completamento di ogni task che costituisce la realizzazione di un estrattore, per una metodologia e per l'altra. A partire da queste considerazioni, nella parte finale del sesto capitolo, attraverso la break even analysis, si è riusciti a individuare i casi in cui risulta economicamente conveniente sviluppare i flussi ETL con i programmi in PL/SQL e quelli in cui è preferibile, per ragioni di efficacia, l'acquisto e l'utilizzo del software. In particolare, è stato evidenziato che per progetti caratterizzati da un elevata complessità, ovvero quelli in cui è prevista la realizzazione di un numero elevato di estrattori (dai 75-80 in su), l'utilizzo del software consente di ammortizzare i costi fissi di acquisto della licenza e di ottenere dei benefici economici rispetto all'utilizzo della programmazione tradizionale, utilizzando la quale, invece, andrebbero realizzati i tipi

di progetto di media o bassa complessità. Un ulteriore fattore di misura della complessità è quello legato al tipo di architettura del contesto applicativo: maggiori sono le fonti sorgenti dei dati tanto più sarà conveniente l'utilizzo del prodotto rispetto allo sviluppo degli estrattori specifici.

Bibliografia

[Albano 2003] A.Albano: *Basi di dati di supporto alle decisioni*, 2008

[Dotnethell 2009] I Dataflow - Le trasformazioni, <http://blogs.dotnethell.it/suxstellino/I-DataFlow-le-trasformazioni__12339.aspx>, 17 febbraio 2009

[Gartner 2008] Gartner, *Magic quadrant for data integration tools*, 2008

[Oracle 2006] Oracle, *The Oracle Data Integrator Architecture*, 2006

[Wikipedia 2009] PL/SQL, <http://it.wikipedia.org/wiki/PL/SQL>, 20 febbraio 2009

Elenco delle figure

Elenco dei listati